Efficient Unbiased Rendering using Enlightened Local Path Sampling

Kristensen, Anders Wang

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Efficient Unbiased Rendering using Enlightened Local Path Sampling

Anders Wang Kristensen

Kongens Lyngby 2010
IMM-PHD-2010-240

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

Summary

Most global illumination algorithms today solve the light transport problem using Monte Carlo ray tracing. These algorithms are capable of producing photo-realistic imagery and in addition have few limitations with respect to the kind of input (geometry, reflection models, etc.) they support. The downside to using these algorithms is that they can be slow to converge. Due to the nature of Monte Carlo methods, the results are random variables subject to variance. This manifests itself as noise in the images, which can only be reduced by generating more samples.

The reason these methods are slow is because of a lack of effective methods of importance sampling. Most global illumination algorithms are based on local path sampling, which is essentially a recipe for constructing random walks. Using this procedure paths are built based on information given explicitly as part of scene description, such as the location of the light sources or cameras, or the reflection models at each point.

In this work we explore new methods of importance sampling paths. Our idea is to analyze the scene before rendering and compute various statistics that we use to improve importance sampling. The first of these are adjoint measurements, which are the solution to the adjoint light transport problem. The second is a representation of the distribution of radiance and importance in the scene. We also derive a new method of particle sampling, which is advantageous compared to existing methods. Together we call the resulting algorithm englightened local path sampling and demonstrate how the algorithm improves efficiency in some hard scenes.

Resumé

I dag løser de fleste global illuminations algoritmer lystransportproblemet ved hjælp Monte Carlo ray tracing. Disse algoritmer er i stand til at levere fotorealistiske billeder og har desuden kun få begrænsninger i forhold til hvilke typer input de understøtter (geometri, refleksionsmodeller, osv.). Ulempen ved disse algoritmer er at de kan være langsomme til at konvergere. Grundet Monte Carlo metodernes natur er resultaterne behæftet med varians, hvilket viser sig som støj i billedet, som kun kan reduces ved at tage flere samples.

Grunden til at metoderne er langsomme skyldes manglen på effektiv importance sampling. De fleste global illuminations algoritmer er baseret på local path sampling, hvilket er en metode til at foretage random walks. Denne metode benytter som udgangspunkt kun information givet eksplicit som del af scene beskrivelsen, som f.eks. positionen af lyskilder eller kameraer, eller refleksionsmodellen i et givet punkt.

I dette projekt udforsker vi nye importance sampling metoder. Vores idé er at analysere scenen før den egentlige rendering finder sted og beregne forskellige statistikker som vi benytter til at forbedre importance sampling. Den første af disse kalder vi "adjoint measurements," da disse er løsningen til det adjunkte lystransportproblem. Det andet er en repræsentation af af radiance og importance i scenen. Vi præsenterer også en nye metode til partikel sampling som har færre ulemper end eksisterende metoder. Samlet set kalder vi metoden "enlightened local path sampling," og demonstrerer at metoden øger effektiviteten i nogle svære scener.

Preface

This thesis was prepared at the Image Analysis and Computer Graphics group at DTU Informatics and submitted to the Technical University of Denmark in partial fulfillment of the requirements for acquiring the Ph.D. degree in informatics and mathematical modeling.

The topic of this thesis is physically based rendering, which is the art and science of creating photo-realistic synthetic images of three-dimensional scenes using a computer. Choosing this topic was not hard, since I have long had a fascination with realistic image synthesis. This fascination stems from a desire to do applied scene, especially applied mathematics and physics. Physically based rendering is a great example of both of these fields in action, together with other fields, such as computer science.

There is something satisfying about writing rendering software that solves the equations that govern the behavior of light, because you can literally see that the science works: the resulting images are similar to the well known visual sensations that come from observing the world. Sometimes the images even predict effects that you were not aware of and that you can confirm exist in the real world. All of this makes rendering a fun and interesting activity.

In fact, writing rendering software can even be a bit addictive. As you sit during long nights of coding and debugging, struggling to implement some rendering algorithm, you suddenly find (what you think is) the last bug. After fixing the error, you rerun the algorithm and suddenly a beautiful image will begin to appear through the Monte Carlo noise. This can be a very rewarding experience.

The downside to rendering is that it requires a great deal of patience. Despite

great improvements in the speed of processors, the time it takes to render scenes with realistic lighting effects is still measured in hours, days, or even longer in some cases. This is a serious problem, since it hinders wider adaption of these techniques. So, in this thesis we will try to investigate new ways of improving the speed of these methods.

Kgs. Lyngby, September 2010

Anders Wang Kristensen

Acknowledgments

First and foremost I would like to thank my family for their support: My parents, Hanne and Arne, and my sisters, Lene and Mette and their husbands Uffe and Jens. And of course my seven nieces and nephews, Christian, Emilie, Kathrine, Caroline, Jacob, Kasper, and Andreas. Without all you guys I would certainly not have been able to complete this thesis.

Secondly, I would like to express my gratitude to my Ph.d. advisor Niels Jørgen Christensen. Niels Jørgen has been a tremendous help in keeping my Ph.d. project on the right track. Whenever I got lost in details, Niels Jørgen would remind me not to lose sight of the big picture. It has also been fun and challenging to help Niels Jørgen teach classes on computer graphics at DTU and even help organize a new course on advanced rendering techniques.

I would also like to thank Jeppe Frisvad for proofreading early chapters of this thesis. Jeppe caught countless spelling errors and his suggestions generally helped clarify the exposition of the thesis. Also my thanks to Andreas Bærentzen for many interesting conversations on geometry and other subjects.

Also, a big thanks to my running buddies from Dyrehaven Motionsklub. Running with this group through rain, snow, and even occasionally sunny weather, has been a big help in clearing my mind after long days in front of my computer.

Finally, I would like to thank our group secretary Eina for always greeting me with a smile and for always being helpful with practical matters.

Symbols and Notation

Mathematical notation

PDF	Probability density function
CDF	Cumulative density function
MSE	Mean square error
RMSE	Root mean square error
X	A random variable
$\mathrm{E}[X]$	The expected value of a random variable
$\mathrm{Var}[X]$	The variance of a random variable
σ/σ^2	The standard deviation / the variance
$\mathbb{R}^n$	The n-dimensional space of real numbers
$\mathcal{V}$	The subset of $\mathbb{R}^3$ defined as the scene
$\partial\mathcal{V}$	The surfaces that form the boundaries in the scene
$\mathcal{V}_0$	The interior points of the scene ($\mathcal{V}_0 = \mathcal{V} \setminus \partial\mathcal{V}$)
$\mathbf{x}$	A position in $\mathbb{R}^3$
$\mathbf{n}(\mathbf{x})/\mathbf{n}_\mathrm{g}(\mathbf{x})$	The geometric normal at a point $\mathbf{x} \in \partial\mathcal{V}$
$\mathbf{n}_\mathrm{s}(\mathbf{x})$	The shading normal at a point $\mathbf{x} \in \partial\mathcal{V}$
$\mathcal{S}^2$	The sphere of directions (the unit sphere in $\mathbb{R}^3$)
$\mathcal{H}^2_+$	The positive hemisphere with respect to a given normal
$\mathcal{H}^2_-$	The negative hemisphere with respect to a given normal
$\boldsymbol{\omega}$	A direction in $\mathcal{S}^2$ (a unit vector in $\mathbb{R}^3$)
$A(D_2)$	The area measure of $D_2 \subset \partial\mathcal{V}$
$V(D_3)$	The volume measure of $D_3 \subset \mathcal{V}_0$
$\sigma(D)$	The solid angle measure of $D \subset \mathcal{S}^2$
$\mathrm{d}A(\mathbf{x})$	Differential area around $\mathbf{x} \in \partial\mathcal{V}$
$\mathrm{d}V(\mathbf{x})$	Differential volume around $\mathbf{x} \in \mathcal{V}_0$
$\mathrm{d}\sigma(\boldsymbol{\omega})$	Differential solid angle around central direction $\boldsymbol{\omega} \in \mathcal{S}^2$

Physics notation

$\mathbf{r}$	A ray $(\mathbf{x}, \boldsymbol{\omega})$
$\bar{\mathbf{x}}$	A path $\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_n$ with $n+1$ vertices (length n)
Ω_k	The space of paths of length k
Ω	The space of paths of any length
$f(\bar{\mathbf{x}})$	The measurement contribution function
$\mu(D)$	The path space measure of $D \subset \Omega$
$\mathrm{d}\mu(\bar{\mathbf{x}})$	Differential path space measure of $\bar{\mathbf{x}} \in \Omega$
ξ	A random number in $[0; 1)$
$\mathbb{U}(0, 1)$	A uniformly distributed random number in $[0; 1)$
t	Time
λ	Wavelength of electromagnetic radiation
$n(\lambda)$	The index of refraction
$\eta(\lambda)$	The real part of the index of refraction
$\kappa(\lambda)$	The imaginary part of the index of refraction
$\sigma_\mathrm{s}(\mathbf{x})$	Scattering coefficient
$\sigma_\mathrm{a}(\mathbf{x})$	Absorption coefficient
$\sigma_\mathrm{t}(\mathbf{x})$	Extinction coefficient
$\tau(\mathbf{x} \leftrightarrow \mathbf{x}')$	The optical depth along the ray segment from $\mathbf{x}$ to $\mathbf{x}'$
$\mathcal{T}(\mathbf{x} \leftrightarrow \mathbf{x}')$	Transmittance between $\mathbf{x}$ and $\mathbf{x}'$
$G(\mathbf{x} \leftrightarrow \mathbf{x}')$	The generalized geometry term
I_j	A measurement
I_j^*	An adjoint measurement
$L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega})$	Incident radiance
$L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega})$	Outgoing (exitant) radiance
$L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega})$	Emitted radiance (volume/surface emission, $L_{\mathrm{e},\mathcal{V}_0} + L_{\mathrm{e},\partial\mathcal{V}}$)
$L_\mathrm{e}^j(\mathbf{x}, \boldsymbol{\omega})$	The jth importance responsivity function
$W_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega})$	Incident importance
$W_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega})$	Outgoing (exitant) importance
$W_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega})$	Total emitted importance
$W_\mathrm{e}^j(\mathbf{x}, \boldsymbol{\omega})$	The jth flux responsivity function
$f_\mathrm{r}(\mathbf{x}, \boldsymbol{\omega}', \boldsymbol{\omega})$	Bidirectional reflection-distribution function (BRDF)
$f_\mathrm{t}(\mathbf{x}, \boldsymbol{\omega}', \boldsymbol{\omega})$	Bidirectional transmittance-distribution function (BTDF)
$f_\mathrm{s}(\mathbf{x}, \boldsymbol{\omega}', \boldsymbol{\omega})$	Bidirectional scattering-distribution function (BSDF)
$f_\mathrm{s}^*(\mathbf{x}, \boldsymbol{\omega}', \boldsymbol{\omega})$	The adjoint BSDF
$f_\mathrm{p}(\mathbf{x}, \boldsymbol{\omega}', \boldsymbol{\omega})$	The phase function
$f_\mathrm{k}(\mathbf{x}, \boldsymbol{\omega}', \boldsymbol{\omega})$	The scattering kernel
$f_\mathrm{k}^*(\mathbf{x}, \boldsymbol{\omega}', \boldsymbol{\omega})$	The adjoint scattering kernel

Contents

Chapter 1

Introduction

A central topic in computer graphics is *rendering*, which can be defined as the process of creating synthetic images from a model using a computer. The term rendering has a long history in computer graphics and has been used to describe tasks as diverse as anything from creating simple wireframe illustrations to drawing complex user interfaces.

Physically based rendering is a form of rendering where the goal is to create physically accurate, or photo-realistic, images of three-dimensional scenes. What differentiates physically based rendering from other types of rendering is that the models used are based on the laws of physics, rather than some ad hoc laws developed specifically for computer graphics.

Physically based rendering can be seen as being one part of the larger task of *realistic image synthesis*, which is the entire process of creating photo-realistic imagery from creating the model to the final result. Realistic image synthesis can be seen as consisting of four distinct parts: The first is a modeling step, where the scene is specified. In the second step the scene is rendered by simulating the behavior of light using the laws of physics. The third step is image reproduction, where the result of the rendering step is mapped to the desired display device. Finally, the last step is a user evaluation of the results.

Modeling of the scene A scene is created by specifying the relative positions and orientations of the objects that comprise the scene. This is usually done using a modeling program or by directly measuring a real scene using a 3D scanner. The shapes of the objects in the scene can be described mathematically using a variety of methods, such as implicit surfaces, polygonal meshes, or higher order surfaces. The objects must be assigned materials, which describe their optical properties, such as how they absorb and scatter light. Some objects also emit light themselves and thereby act as the *light sources* of the scene. It is also necessary to include a specification of the *sensors* in the scene. Sensors, the most common kind of which is a camera, define where in the scene we are interested in measuring the light.

Physically based rendering Physically based rendering is the task of computing how light emitted from light sources is scattered by the objects in the scene and impinge on the sensors. The goal is to compute a set of *measurements*, which are simply numerical values which describe the response of the sensors to incoming light energy. Most often the sensors model a camera, and in this case the measurements correspond directly to the pixel values of the resulting image. To perform the measurements it is necessary to solve the mathematical equations that govern the physics of light.

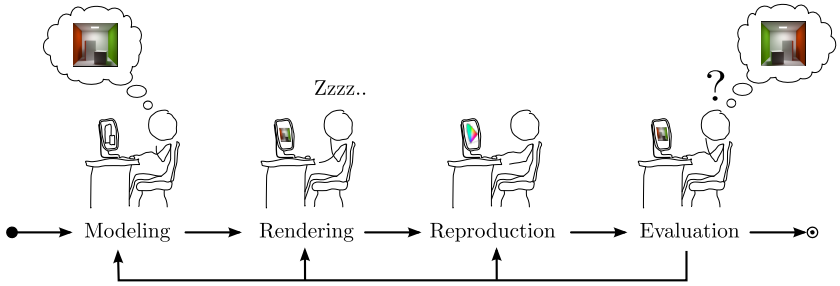
Image reproduction The result of the rendering step is a set of measurements. In some cases the numerical values of the measurements are the end result and no further action is needed. More often the result is intended for human viewers to be seen on display device, such as a computer monitor or on a hard copy from a printer. In this case, the goal of the reproduction step is to map the measurements to the display device in such a way that observing the synthetic image on the display device evokes the same response that the viewer would have gotten from watching a realization of the virtual scene. This is obviously a very lofty goal, which can be very challenging to reach due to the limited technology of current display devices, such as restricted dynamic range and limited color gamut. It is further complicated by the nonlinearities of the human visual system.

Evaluation The last step is evaluation of the results of the image reproduction step. Here it is determined whether the results are satisfactory or if any adjustments to the process are required.

Based on the above description, it should be clear that realistic image synthesis is more than just physics. Instead, as illustrated in Figure 1.1, it is an iterative process. First the user models and renders an image of the scene, which is then reproduced. The resulting image is evaluated, which may cause the user to go

back and fine-tune the parameters that control the image reproduction step, adjust how rendering is performed, or even go back and make adjustment to the scene. This process is then repeated until the desired result is achieved.

Figure 1.1: Realistic image synthesis is the entire process of creating photo-realistic synthetic images. This includes creating the model of the three-dimensional scene, rendering an image of the scene, and reproducing that image on a display device. It is an iterative process, where the parameters that control different stages are tweaked until the desired result is achieved.



The quality of the end result will be limited by the weakest link in this process. For instance, if the scene is poorly modeled, then no matter how accurately the physics of light is simulated, the results will always end up appearing unrealistic. Similarly, even if the model of the scene is precise and light is simulated accurately, the results can still be disappointing if the image reproduction step is poorly executed or if the display device is too limited to faithfully reproduce the scene.

There are several application areas for realistic image synthesis. The best known of these is the entertainment industry, where realistic image synthesis is used for creating special effects in movies and also in computer games. Realistic image synthesis is also used increasingly for predictive rendering (see e.g. Wilkie, Weidlich, Magnor, and Chalmers [2009]). Predictive rendering is the process of inferring the appearance of an object or scene based on a model. For predictive rendering to be of any use, the simulation of light has to be of a sufficiently high quality, so that the results not only look real, but are in fact reliable predictions of the actual appearance of the finished product. Predictive rendering has obvious applications in architectural design, where buildings can be visualized prior to construction. Predictive rendering is also used in product design to reduce development costs and time to market. Cost can be reduced by relying less on the time-consuming construction of expensive prototypes, and instead using predictive rendering tools to evaluate design alternatives. The same techniques that underlie physically based rendering can also be used for lighting design and lighting engineering. Lighting design is the process of determining the best

configuration of light sources and the materials in e.g. an office, so as to create a comfortable lighting environment, whereas lighting engineering is the design of the lighting fixtures themselves. Both of these tasks require simulating the behavior of light, similarly to physically based rendering.

All the steps involved in realistic image synthesis are challenging subjects worthy of investigation. However, in order to limit the scope of this work, we will not consider the process of modeling the scene any further, but instead simply assume that scenes of the appropriate kind are available as input to the rendering step. We will also not consider the image reproduction step any further. Instead, we will focus only on the rendering step.

1.1 Physically Based Rendering

As discussed above, physically based rendering is concerned with simulating the behavior of light with the goal of producing photo-realistic synthetic images intended for human viewers. Light is electromagnetic radiation, and in this thesis we will use the word "light" exclusively to mean visible light, i.e. the part of the spectrum of electromagnetic radiation that is visible to humans.

The history of light is also the history of some of the most important discoveries in physics over the last few centuries. Many increasingly more accurate models of light have been suggested over the years, and as a result the most sophisticated models of light can today predict virtually any observed phenomenon. Unfortunately, as these models became more accurate, they also became more complicated, and thus more computationally intensive. This means that when choosing a suitable model of light, a compromise has to be made. On the one hand, we must choose a model that is rich enough to be able to predict the effects we are looking for. On the other hand, we are always interested in picking the simplest model, since implementation of the resulting algorithm will be simpler and computations less time consuming.

Physically based rendering is mostly concerned with the large-scale behavior of light as it interacts with matter. By "large-scale," we mean interaction with matter on a scale much larger than the wavelength of visible light and over time periods large compared to the frequency of visible light. This process is known as *light transport* and in computer graphics the process of determining the appearance by simulating light transport is known as *global illumination*.

Geometrical optics, also called ray optics, accounts for the large-scale behaviors of light, such as linear propagation, reflection, and refraction. Geometrical

optics is capable of predicting practically all important optical phenomena, while still being fairly simple, and is therefore the most popular model of light in computer graphics. However, traditional geometrical optics does not include concepts necessary to quantify light energy, so for this theory to be useful it has to augmented with radiometric concepts such as radiance.

Physical optics is a more advanced theory and includes all the effects of geometrical optics and adds interference, diffraction, dispersion, and more. Physical optics is much more complicated than geometrical optics and the additional phenomena this theory predicts are of little importance in the majority of scenes, so using it for global illumination is therefore rarely justified. However, physical optics and other more advanced theories are still useful in computer graphics to infer macroscopic properties used as part of geometrical optics (this field is known as *appearance modeling*; see e.g. Frisvad [2008]).

The behavior of light can also be studied at the transport level. *Radiative transfer* uses geometrical optics and conservation laws from thermodynamics to describe the distribution of radiant energy. The resulting model can predict phenomena such as participating media, e.g. fog or smoke, and subsurface scattering, scattering at surfaces, and much more. This model is presently the most popular in computer graphics and is also the model used throughout this work.

1.2 Existing Methods

Global illumination algorithms have a more than 40 year long history in computer graphics. Early global illumination algorithms were restricted by the technology available at the time. While many were physically based, they did not account for the full range of optical effects that the underlying model of light predicted, or they were limited with respect to the kinds of geometry or reflection models they supported.

The quality of global illumination algorithms can be judged according to the degree to which they possess certain characteristics. The first of these is the limitations of the algorithms with respect to the kind of scenes they support. Ideally, global illumination algorithms should support any kind of scene, including scenes with area/volume light sources, general camera and reflection models, participating media, and subsurface scattering. However, often a given algorithm only supports a subset of these features or the algorithm supports them but only at the cost of a disproportionate increase in computational resources. A second way of evaluating a global illumination algorithm is according to how accurately the light transport simulation is performed. Ideally, algorithms should account

for all light energy emitted from the light sources that end up at the camera no matter how complicated the underlying transport mechanism is. However, presently all global illumination algorithms are limited in some way in this respect, and therefore have missing lighting effects, such as e.g. missing indirect illumination or caustics in the most severe cases. Finally, algorithms can be judged according to the amount of computational resources they require. This includes processing time, but also memory and other resources. A related issue is the robustness of the algorithm. For some algorithms making even small changes to a scene, such as adding a mirror, can greatly increase computational burden. Such algorithms are not robust and should be avoided.

In the following we will give a brief overview of the most important developments in the history of global illumination algorithms. Global illumination algorithms can be classified according to their solution space. Of course, the goal of any global illumination algorithm is to produce an image of a scene from particular view point, so image space is the final solution space for all of these algorithms. However, *view-independent* algorithms first produce a global solution to the light transport problem without considering the view point. This solution can then be visualized from a particular view point often even interactively. Such algorithms are always limited to scenes with low-frequency (diffuse) reflection models, since both computing and storing a global solution for general reflection models is not practical. Some view-independent algorithms produce a global solution where the amount of detail in the different parts of the solution space depends on how important those parts are to a given view point or set of view points. Such algorithms are called *importance-driven*, and have the advantage that because the global illumination computations are focused on the important parts of the scene, the quality of the final result is improved for the same amount of computational resources. *View-dependent* algorithms come in several different flavors. Multi-pass algorithms first compute a partial global solution using a view-independent method. This solution is then used in a view-dependent pass, where high-frequency (specular or mirror-like) reflection is added. The fundamental limitation of these algorithms is that they often only partially account for the interaction between the high and low-frequency components of light transport. Finally, image space algorithms directly compute the values at each pixel without resorting to any intermediate step. These methods are often conceptually simpler, but can be less effective.

Most global illumination algorithms today are based on *ray tracing*. Ray tracing is essentially a geometrical optics approach, where rays are followed along straight lines starting from the eye. As the rays traverse the scene they are scattered and change direction instantaneously and form *paths* through the scene. *Full paths* that connect the eye and light sources can be constructed e.g. by connecting vertices from the paths to points on the light sources. Based on the geometry of the path the amount of flux transferred along the path can be

computed, which directly leads to the formation of an image of the scene.

A restricted form of ray tracing, called ray casting, was first used Appel [1968] to render shaded line drawings. This work was later extended by Whitted [1980], who is the person usually credited with introducing ray tracing to the computer graphics community. This early form of ray tracing, nowadays referred to as classical ray tracing, is based on a recursive algorithm that terminates at diffuse surfaces, where lighting from point light sources is evaluated using a shading model, but continues the recursion at specular surfaces. The advantage of this formulation is that the need for time-consuming numerical integration is conveniently sidestepped, though at the price of not supporting diffuse interreflection and other lighting effects. A consequence of this is that images produced with classical ray tracing have a distinctive computer-generated look.

Finite element methods have also been used as the basis for global illumination algorithms. The *radiosity method* by Goral, Torrance, Greenberg, and Battaile [1984] uses techniques borrowed from thermal engineering to compute a global solution which includes diffuse interreflection. In order to use the radiosity method it is necessary to discretize the scene. This has turned out to be the Achilles' heel of the method, since finding suitable basis functions that work for general scenes remains an unsolved problem. Much research has been devoted to this problem, which has led to algorithms such as substructuring [Cohen, Greenberg, Immel, and Brock, 1986], progressive refinement [Cohen, Chen, Wallace, and Greenberg, 1988], and discontinuity meshing [Lischinski, Tampieri, and Greenberg, 1992] to name a few. Importance-driven methods have also been used with the radiosity method by Smits, Arvo, and Salesin [1992], who made the accuracy of the different parts of the global solution depend on the view point. The radiosity method has also been extended to non-diffuse surfaces by Immel, Cohen, and Greenberg [1986] among others and to participating media by Rushmeier and Torrance [1987]. However, despite a tremendous amount of research, the radiosity method is today still not capable of handling general scenes, such as scenes with significant high-frequency (non-diffuse) reflection and participating media, and is therefore falling into disuse.

Algorithms based on Monte Carlo methods are presently the most popular way of solving the light transport problem. Monte Carlo methods are a set of algorithms for solving hard integration problems using random sampling. The primary advantage of these methods is their generality: If combined with ray tracing these methods support practically any kind of reflection model, geometry, light source type, or sensor type, as long as these are based on point sampling, which is a very mild restriction. Monte Carlo methods can be classified according to whether they are unbiased or biased but consistent. Briefly stated, unbiased methods compute the correct answer on average, whereas biased but consistent methods produce the correct answer in the limit. An advantage of un-

biased methods is that they allow error estimates, unlike biased methods, where it is often not even possible to bound the bias. This makes unbiased methods desirable when the goal is to compute accurate measurements, i.e. measurements that we can be confident are correct, which is necessary in e.g. predictive rendering. However, often the goal of realistic image synthesis is simply to produce pleasing pictures, rather than accurate measurements, and in this case biased but consistent methods can be used and are often much more efficient.

Classical ray tracing lacks support for lighting effects such as penumbra from area light sources, depth of field, motion blur, and indirect illumination. The distribution ray tracing algorithm by Cook, Porter, and Carpenter [1984] uses random sampling, to compute some of these "fuzzy phenomena." Random sampling, which is essentially a Monte Carlo method, was also used in the seminal paper by Kajiya [1986]. In this paper Kajiya presented the "The Rendering Equation," which is an integral equation that describes the distribution of radiance in a scene without participating media. This equation is significant, since this is the equation all global illumination algorithms attempt to solve. It can therefore be used to compare these algorithms according to how accurately they do so. Kajiya also recognized that the rendering equation could be solved by transforming it into an integration problem and suggested an unbiased Monte Carlo method based on random walks, called *path tracing*, for doing so.

Path tracing works well for diffuse interreflection, but has a hard time finding light paths that produce caustics. Conversely, the *light tracing* algorithm by Dutré, Lafortune, and Willems [1993], which is based on random walks starting from the light sources, is relatively efficient for caustics paths, but less so for direct illumination. The insight that some paths are easier to find if the random walks are started from the light sources and others if the walk is started from the eye led to the development of *bidirectional methods*. An early example of a bidirectional method is the algorithm by Arvo [1986]. This is a two-pass method, where the first pass is spent tracing rays from the light sources and accumulating light energy in textures across the surfaces of the objects in the scene. In the second pass the scene is rendered from the eye using the light energy stored in the texture maps. A more popular variation of this approach is the *photon mapping* algorithm by Jensen [2001]. In this algorithm rays, or "photons," are also emitted from light sources, though rather than storing them in texture maps, they are instead stored in a *kd*-tree. The final image is rendered using ray tracing starting from the eye, where radiance is computed using density estimation based on the photons. Algorithms based on *virtual point lights*, such as the instant radiosity algorithm by Keller [1997], follow a similar approach. In those algorithms point light sources that represent the illumination in the scene are created using random walks starting from the light sources. The final image is created using ray tracing from the eye, where the incident illumination is estimated using the point light sources. Common for

these algorithms is that they gain much of their efficiency from path reuse, i.e. the representation of illumination is created once but reused for all the pixels in final image. The downside to using such intermediate representations is that their quality may not be sufficiently high to reproduce all lighting effects; e.g. non-diffuse reflection can be difficult. Another issue is that due to reuse, error is correlated between the pixels, which can lead to objectionable artifacts in the final result. Not all bidirectional methods require intermediate storage. *Bidirectional path tracing*, which was developed independently by Lafortune and Willems [1993] and Veach and Guibas [1994, 1995], computes pixel intensities directly similarly to path tracing and light tracing. This is achieved by forming paths using a pair of random walks, one starting from the light source and one starting from the sensor.

The random walks used in the aforementioned algorithms are all created using the principle of *local path sampling*, which also forms the basis of most other ray tracing based global illumination algorithms. The principles that underlie these algorithms was studied in great detail in the Ph.D. thesis of Eric Veach [1997], who gave the theory a solid mathematical foundation. In particular Veach studied the symmetry between radiance and importance in a scene and gave a precise mathematical description of the equilibrium distribution of importance. This important work is quoted numerous times throughout this thesis. Veach also investigated the limitations of local path sampling, and suggested the *Metropolis light transport* algorithm [Veach and Guibas, 1997] for dealing with difficult lighting situations, i.e. scenes where local path sampling is ineffective. This algorithm is still based on local path sampling, but is framed in a Markov Chain Monte Carlo context, where paths are explored by mutating a Markov chain using the Metropolis-Hastings algorithm.

Importance-driven methods have also been used to improve local path sampling. For instance, Dutré and Willems [1994] suggest an adaptive algorithm for improving light tracing by learning which outgoing directions on the light sources that are likely to lead paths to areas visible to the camera. Similarly, Jensen [1995] suggests improving path tracing by using the photon map to guide paths to light sources (confusingly, the author calls this importance-driven path tracing; a more appropriate description of this technique would be radiance-driven path tracing).

Ray tracing has also been performed on graphics processing units (GPUs) by Carr, Hall, and Hart [2002] and Purcell, Buck, Mark, and Hanrahan [2002] among others. GPUs were originally special purpose hardware designed for fast rasterization. However, over the years graphics hardware has become increasingly programmable to the point where GPUs today are practically fully programmable. The lure of using GPUs for physically based rendering is that they offer higher peak performance than regular general purpose processors

(CPUs). However, to achieve this speed it is necessary to express the algorithms in an efficient way using the stream programming model. This has turned out to be very challenging, since ray tracing, unlike rasterization, has highly incoherent memory access patterns.

1.3 Thesis Statement

Today the challenge when solving the light transport problem is no longer that we do not have algorithms that can handle all lighting effects, but rather that these algorithm require too many computational resources.

Bidirectional path tracing and Metropolis light transport are among the most effective unbiased Monte Carlo ray tracing algorithms today. These algorithms construct paths using local path sampling based on information given explicitly as part of the scene description, such as the location of the cameras, or how much light is emitted from the light sources, or how light is scattered at a given point. This makes it difficult to sample paths effectively in many situations. For instance, random walks starting from the eye are built without considering the distribution of radiance in the scene (except for emitted radiance, which is given explicitly as part of the scene description). Similarly, random walks starting from the light sources do not take importance into account. This leads to inefficiencies when the distributions of radiance and importance in a scene are dissimilar, which is the case in many common lighting situations. For instance, consider an office lit by skylight streaming in through a window. In such cases bidirectional path tracing degenerates to an inefficient form of path tracing, since the light path is essentially wasted, and the Metropolis light transport algorithm struggles with making good mutations.

The goal of this project is to investigate how additional information derived from the scene description can be used to improve local path sampling and thereby increase the effectiveness of these algorithms. We call this additional information *global context*, since it is based on the properties of the entire scene. We consider three different kinds of global context. The first kind is an approximate representation of the equilibrium distribution of radiance in the scene. We use this distribution to guide random walks starting on the sensors to brightly lit areas of the scene. The second kind is a record of how much light energy each part of the light sources contribute to the final image. We call this data *adjoint measurements* and use it to pick good starting points for random walks starting on light sources. The last kind we consider is an approximate representation of the equilibrium distribution of importance in the scene. We use this distribution to guide random walks starting at the light sources to the visible parts of the scene. We call the resulting algorithm *enlightened local path sampling* to

differentiate it from regular local path sampling.

The methods presented in this work can be used with any algorithm based on local path sampling. However, to limit the scope of the work, we restrict our attention to the subset of global illumination algorithms that are unbiased. We also only consider designing algorithms for use on CPUs, and do not consider the challenges of mapping the proposed algorithms to GPUs. Finally, many implementation details that are necessary when designing rendering software, such as acceleration structures, are not covered in this work. We refer the reader to the book by Pharr and Humphreys [2004] for information on these topics.

1.4 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2: The Light Transport Problem

We begin with a description of the light transport problem as it is usually posed in computer graphics. Starting from transport theory we formulate the equation of transfer with boundary conditions, which describes the distribution of particles in an abstract setting. We then apply this theory to the transport of radiant energy, which gives rise to the fields radiative transfer and radiometry. We show how to provide a mathematical description of the scenes that are the input to the rendering algorithms. This description includes how to define scattering at boundaries, including the challenges of asymmetric scattering, and how to define volume scattering. We also include a description of how to define emission from light sources and how to define the sensors in the scene. Finally we present the measurement equation, which binds together the equation of transfer, the scene description, and the description of the sensors, and which defines the mathematical solution to the light transport problem.

Chapter 3: Monte Carlo Methods

Chapter 3 provides a general introduction to Monte Carlo methods. The light transport problem is a high-dimensional integration problem that only Monte Carlo methods currently provide a practical way of solving. We also discuss the fundamental advantage of Monte Carlo over other integration schemes, which is that convergence rate is independent of the dimensionality of problem. The remainder of the chapter is concerned with improving the efficiency of Monte Carlo estimators. We present the important variance reduction methods of importance sampling and stratified sampling, among others. We conclude the chapter with an introduction to Markov Chain Monte Carlo based on the Metropolis-Hastings algorithm,

and discuss the challenges of constructing good mutation strategies for this algorithm.

Chapter 4: Solution Strategies

In this chapter we show how to solve the light transport problem using Monte Carlo methods. We begin by presenting the integral formulation of light transport and show how to use linear operators to reason about the existence of solutions to the light transport problem. Adjoint operators are introduced and these allow us to define the useful concept of importance. We then present the path integral formulation of the light transport problem and discuss the important principle of local path sampling. Using this framework we explain the classical unbiased global illumination algorithms. We conclude the chapter with a discussion of the limitations of existing unbiased algorithms. In particular we discuss how effective importance sampling is hampered by the lack of global context.

Chapter 5: Enlightened Local Path Sampling

We begin this chapter with a discussion of how global context can be used to improve local path sampling. To this end we introduce the concept of adjoint measurements. Adjoint measurements are the solution to the adjoint light transport problem, i.e. they measure how important the different parts of the light sources are to the sensors. This makes adjoint measurements a useful tool for selecting good starting points on light sources in bidirectional algorithms. We propose a two-pass algorithm for implementing local path sampling using this global context. In the first pass we analyze the flow of radiance and importance in the scene. This is accomplished using a variation of particle tracing that allow us to sample radiance and importance simultaneously. Using this procedure we compute the adjoint measurements. We also construct a data structure for representing the radiance and importance in scene that is compact, while still being amendable to importance sampling. We call local path sampling using these data structures for "enlightened," since rather than sampling paths blindly, we take adjoint quantities into account. Enlightened local path sampling can be used as a drop-in replacement for any algorithm based on regular local path sampling. We use it with a variation of Metropolis light transport and show that particularly for hard lighting situations the proposed method is an advantage.

Chapter 6: Conclusion

We conclude with a summary of the results and an overview of possible directions for future research.

The bibliography can be found starting on page 171 and the index can be found starting on page 183.

Chapter 2

The Light Transport Problem

Images are formed when photons, that are emitted from light sources and scattered in the scene, hit a sensor, such as a camera or the retina in the eye. The amount of light energy that hits the sensor determines how strongly the sensor responds. We will call the actual numeric value that describes the response of the sensor a measurement, and we will define the light transport problem as the task of computing these measurements for each sensor in the scene.

Computing measurements requires a model of how light behaves. Using this model, we can describe how light is emitted in the scene and how sensors respond to light. In order to perform these computations, it also necessary to have an understanding how light interacts with matter. This means that the description must also include information on how the materials of the objects in the scene cause light to scatter.

In this chapter we provide a mathematical description of the light transport problem suitable for global illumination algorithms based on Monte Carlo ray tracing. We begin by studying transport theory. As discussed in the previous chapter, geometrical optics, which is essentially a particle model of light, is a suitable model of light for global illumination. Transport theory, is the study of such particles and uses conservation laws from thermodynamics to derive an equation that governs their distribution. We apply transport theory to the transfer of radiant energy, which leads to radiative transfer. Using radiative

transfer we can describe the distribution of radiant energy in a scene using radiometric concepts such as radiance and distribution functions. We then discuss the requirements for a complete scene description, including a how to describe the light sources, the sensors, and how to describe scattering. We conclude by presenting the measurement equation, which links together all these elements and defines the solution to the light transport problem.

2.1 Domains and Measures

Before we begin, it will be necessary to introduce some mathematical concepts, such as solid angle, as well as the domains and measures that we will use.

Much of this chapter is spent considering the distribution of light energy in a 3D scene. In this regard, we will use $\mathcal{V}$ to mean the finite subset of $\mathbb{R}^3$ that is of interest. The points that belong to $\mathcal{V}$ can be classified into interior points, $\mathcal{V}_0$, and boundary points, $\partial\mathcal{V}$, so that $\mathcal{V} = \mathcal{V}_0 \cup \partial\mathcal{V}$. The *area measure* is denoted $A(D_2)$, where $D_2 \subset \partial\mathcal{V}$. Similarly, the *volume measure* is denoted $V(D_3)$, where $D_3 \subset \mathcal{V}_0$.

Directions are denoted $\boldsymbol{\omega}$ and we use $\mathcal{S}^2$ to specify the set of all directions (the unit sphere in $\mathbb{R}^3$). We will use $\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{x}'} = \frac{\mathbf{x}'-\mathbf{x}}{\|\mathbf{x}'-\mathbf{x}\|}$ to mean the unit vector from $\mathbf{x}$ to $\mathbf{x}'$, where $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^3$ and $\|\cdot\|$ is the L_2-norm. We will use σ to mean the surface area measure on $\mathcal{S}^2$, which we call the *solid angle measure*. This means that if $D \subset \mathcal{S}^2$ then $\sigma(D)$ is the solid angle of D. The solid angle is simply the 3D analog to the more familiar 2D concept of angle. For instance, the solid angle of the entire sphere of direction is $\sigma(\mathcal{S}^2) = 4\pi$. The solid angle subtended by some object seen from a point $\mathbf{x}$ can be found by projecting the object onto a unit sphere centered at $\mathbf{x}$ and then computing the area covered by the resulting shape (see Figure 2.1a).

The solid angle measure is related to the area measure by

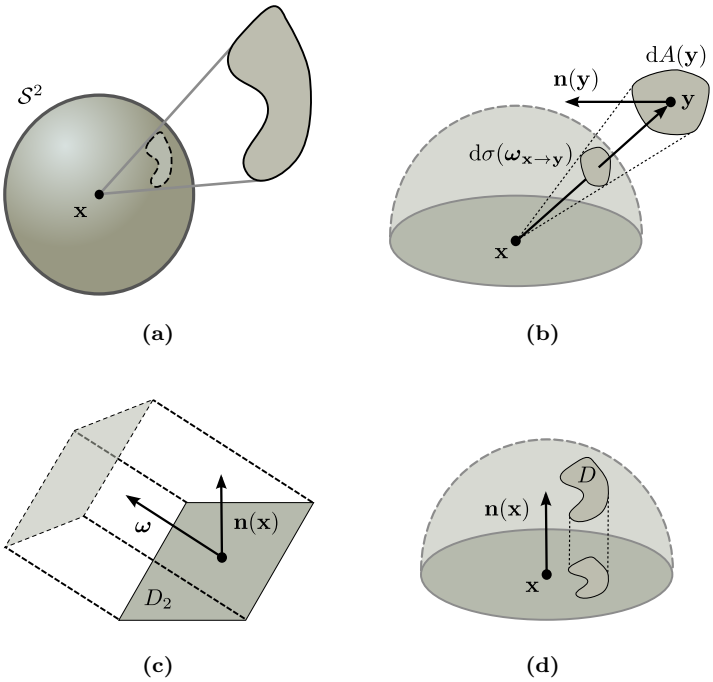
$$\mathrm{d}\sigma(\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}) = \frac{|\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}} \cdot \mathbf{n}(\mathbf{y})|}{\|\mathbf{x}-\mathbf{y}\|^2}\ \mathrm{d}A(\mathbf{y}),$$

where $\mathbf{n}(\mathbf{y})$ is the *normal* at $\mathbf{y} \in \partial\mathcal{V}$ (see Figure 2.1b). This relation is necessary to account for the change of variables that occurs when changing the domain of an integral from the solid angle measure to the area measure.

It will also be convenient to introduce the *projected area measure*,

$$A^{\perp}_{\boldsymbol{\omega}}(D_2) = \int_{D_2} |\boldsymbol{\omega} \cdot \mathbf{n}(\mathbf{x})|\ \mathrm{d}A(\mathbf{x}).$$

Figure 2.1: The solid angle (top-left) of an object seen from $\mathbf{x}$ is found by first projecting it onto a unit sphere centered at $\mathbf{x}$ and then computing the dashed area. The relationship between differential solid angle and differential area is shown in the top-right figure. Projected area (bottom-left) is found by projecting the shape onto a plane along the direction $\boldsymbol{\omega}$. Projected solid angle (bottom-right) is found by projecting onto the tangent plane defined by the normal.



Similarly, *projected solid angle measure* is defined as

$$\sigma^{\perp}_{\mathbf{x}}(D) = \int_D |\boldsymbol{\omega} \cdot \mathbf{n}(\mathbf{x})| \; \mathrm{d}\sigma(\boldsymbol{\omega}).$$

Refer to Figure 2.1c and Figure 2.1d for a more intuitive explanation of these concepts. Consequently, we have the relationships

$$\begin{aligned} \mathrm{d}A^{\perp}_{\boldsymbol{\omega}}(\mathbf{x}) &= |\boldsymbol{\omega} \cdot \mathbf{n}(\mathbf{x})| \; \mathrm{d}A(\mathbf{x}) \quad \text{and} \\ \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\boldsymbol{\omega}) &= |\boldsymbol{\omega} \cdot \mathbf{n}(\mathbf{x})| \; \mathrm{d}\sigma(\boldsymbol{\omega}). \end{aligned}$$

In order to simplify notation, it will be useful to define a pair of measures over

all of $\mathcal{V}$, where the type of the measure depends on whether the subset belongs to $\mathcal{V}_0$ or $\partial\mathcal{V}$. The first of these is the area/volume measure,

$$\mathrm{d}\lambda(\mathbf{x}) \equiv \begin{cases} \mathrm{d}A(\mathbf{x}) & \text{if } \mathbf{x} \in \partial\mathcal{V}, \\ \mathrm{d}V(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{V}_0. \end{cases}$$

The second is the projected area/volume measure,

$$\mathrm{d}\alpha_{\boldsymbol{\omega}}(\mathbf{x}) \equiv \begin{cases} \mathrm{d}A^{\perp}_{\boldsymbol{\omega}}(\mathbf{x}) & \text{if } \mathbf{x} \in \partial\mathcal{V}, \\ \mathrm{d}V(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{V}_0. \end{cases}$$

2.2 Transport Theory

The study of the behavior of "particles" in an abstract setting is called *transport theory*. The particle model of light is well suited for simulating light flow in a computer graphics context, so in our case the particles can be though of as photons. When transport theory is applied to photons, as opposed to say neutrons or gas molecules, it gives rise to the field of *radiative transfer*.

Radiative transfer, described in such famous books as Chandrasekhar [1960] and Preisendorfer [1965], is the study of the interaction of electromagnetic radiation with matter on a macroscopic scale. The origin of the field of radiative transfer can be found in the seminal papers of Schuster [1905] and Schwarzschild [1906]. Arvo [1993], Cohen and Wallace [1993], and Glassner [1994] all give presentations of transport theory and radiative transfer in the context of computer graphics, that we will follow here.

Our goal will be to find a balance equation, called the *equation of transfer*, which describes the equilibrium distribution of particles in a scene. As we will see, the equation of transfer is a integro-differential equation, called the Boltzmann equation. In the following, the equation of transfer for particles in the abstract setting will be derived. In the next section these equations will be applied to the transport of radiant energy which will give rise to the fields of radiative transfer and radiometry.

2.2.1 Equation of Transfer

The purpose of transport theory is to find the distribution of particles in space and time given a medium with suitable boundaries, where the resulting distribution is expressed as geometrical and physical properties of the host medium.

In order to find expressions for these distributions, it is necessary to first crystallize what is meant by an abstract particle and what assumptions underlie this particle model.

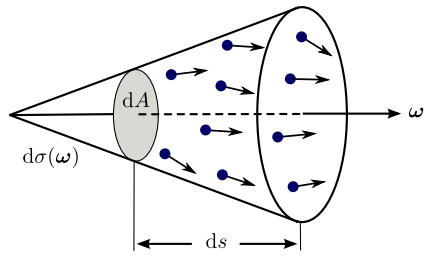
Particles in transport theory are described by their position and velocity. In addition, particles can have internal state, such as frequency (or wavelength), and polarization. The complete state of particle can be seen as a point in the particles' *phase space*.

Particles are assumed to be infinitely small and their number is assumed to be so large, that their distribution can be treated as continuum. It is assumed that particles do not interact, i.e. they never collide or attract each other and thus only interact with the medium. In addition, the particles are not influenced by external forces. Finally, it is assumed that all collisions with the medium and boundaries are completely elastic, and that all particles move with the same speed, v.

One-speed transport theory has a five dimensional phase space: $\mathbb{R}^3 \times \mathcal{S}^2$. To describe how many particles that have a particular configuration, we treat the particles as a continuum, and talk about the *phase space density*, $n(\mathbf{x}, \boldsymbol{\omega}, t)$, of the medium. The number of particles in a differential volume, $\mathrm{d}V(\mathbf{x})$, around $\mathbf{x} \in \mathbb{R}^3$ moving in directions within a differential solid angle of $\mathrm{d}\sigma(\boldsymbol{\omega})$ around the central direction $\boldsymbol{\omega} \in \mathcal{S}^2$ at time t is then given by

$$n(\mathbf{x}, \boldsymbol{\omega}, t)\ \mathrm{d}V(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega}).$$

Figure 2.2: Phase space flux is the rate at which particles cross a surface per unit area per unit solid angle. The number of particles in the volume $\mathrm{d}A\ \mathrm{d}s$ is $n(\mathbf{x}, \boldsymbol{\omega}, t)\ \mathrm{d}A(\mathbf{x})\ \mathrm{d}s\ \mathrm{d}\sigma(\boldsymbol{\omega})$, or, in terms of flux, $\phi(\mathbf{x}, \boldsymbol{\omega}, t)\ \mathrm{d}A(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega})\ \mathrm{d}t$.



It turns out to be convenient to consider the rate at which particles cross a possible imaginary surface. As illustrated in Figure 2.2, if the surface has dif-

ferential area $\mathrm{d}A$, then the resulting volume is $\mathrm{d}A(\mathbf{x})\ \mathrm{d}s$, where $\mathrm{d}s = v\ \mathrm{d}t$, and the number of particles in this volume is

$$n(\mathbf{x}, \boldsymbol{\omega}, t)\ \mathrm{d}A(\mathbf{x})\ \mathrm{d}s\ \mathrm{d}\sigma(\boldsymbol{\omega}).$$

The notion of flow across a surface can also be expressed in terms of the *phase space flux*

$$\phi(\mathbf{x}, \boldsymbol{\omega}, t) \equiv v\, n(\mathbf{x}, \boldsymbol{\omega}, t),$$

in which case the number of particles in the differential volume $\mathrm{d}A(\mathbf{x})\ \mathrm{d}s$ is

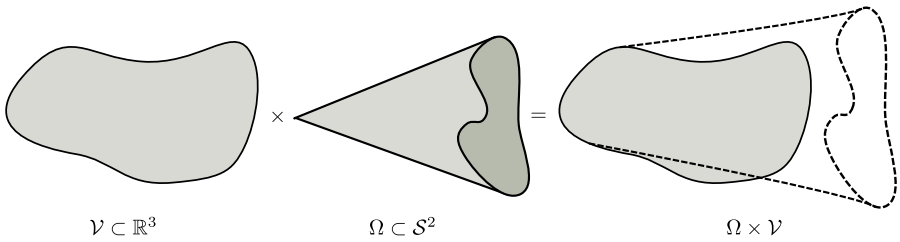
$$\phi(\mathbf{x}, \boldsymbol{\omega}, t)\ \mathrm{d}A(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega})\ \mathrm{d}t.$$

Consider the subset of the phase space, $\mathcal{V} \times \Omega$, where $\mathcal{V} \subset \mathbb{R}^3$ and $\Omega \subset \mathcal{S}^2$, illustrated in Figure 2.3. We will denote the surface that forms the boundary of $\mathcal{V}$ as $\partial\mathcal{V}$ and the interior as $\mathcal{V}_0 = \mathcal{V} \setminus \partial\mathcal{V}$ and in the following assume that $\mathbf{x}_\mathrm{s} \in \partial\mathcal{V}$ and $\mathbf{x}_\mathrm{v} \in \mathcal{V}_0$. Then, the number of particles in the subset $\mathcal{V} \times \Omega$ is

$$N(t) = \int_\Omega \int_{\mathcal{V}_0} n(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}, t)\ \mathrm{d}V(\mathbf{x}_\mathrm{v})\ \mathrm{d}\sigma(\boldsymbol{\omega}).$$

There are two ways the number of particles in the volume can change as function of time. Firstly, the medium can have time-dependent properties; e.g. emission can change over time, or the scattering properties might depend on time. Secondly, if any of properties of the medium have recently changed, the medium might be moving toward equilibrium.

Figure 2.3: The subset of phase space $\Omega \times \mathcal{V}$ is defined as the Cartesian product of Ω and $\mathcal{V}$, i.e. $\{(\mathbf{x}, \boldsymbol{\omega}) | \mathbf{x} \in \mathcal{V} \text{ and } \boldsymbol{\omega} \in \Omega\}$.



Fortunately, the time it takes for light to reach equilibrium is very short in environments on a human scale. In fact, light travels so fast compared to the size of macroscopic features, that it is commonplace to assume that the speed of light is infinite and that equilibrium is reached instantaneously. This means that as long as the properties of the medium are undisturbed, the number of particles in the volume $\mathcal{V} \times \Omega$ does not change; i.e.

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = 0.$$

As a consequence, phase space density, and consequently phase space flux, are independent of time and the time index can be dropped.

This is an example of dynamic equilibrium. The number of particles in $\mathcal{V} \times \Omega$ is constant, but particles are still flowing in and out of the volume. Since the number is constant, the number of particles flowing in must exactly match the number flowing out, i.e. gains and losses must be equal. This means we can write down a balance equation for gains and losses, if the sources of these can be identified.

The three factors that influence the number of particles in a given volume are *emission*, *collision*, and *streaming*. Emission is simply the creation of new particles in the volume. This can happen as a result of various physical processes, such as incandescence. The exact nature of these processes is not of concern here; only the state of the generated particles matter.

Collisions of particles with the medium results in either absorption or scattering. If the particle is absorbed, it disappears from the model, and its energy is converted to some other form, such as heat. Scattering is the instantaneous change in direction of a particle. Streaming happens if a particle enter or leaves the volume through the surface.

Emission

Volume emission is described by the *phase space source* function, $q_{\mathrm{v}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega})$. As shown in Figure 2.4, it describes how many particles are generated per unit volume, per unit solid angle, per unit time. The change in N caused by volume emission can be found to be

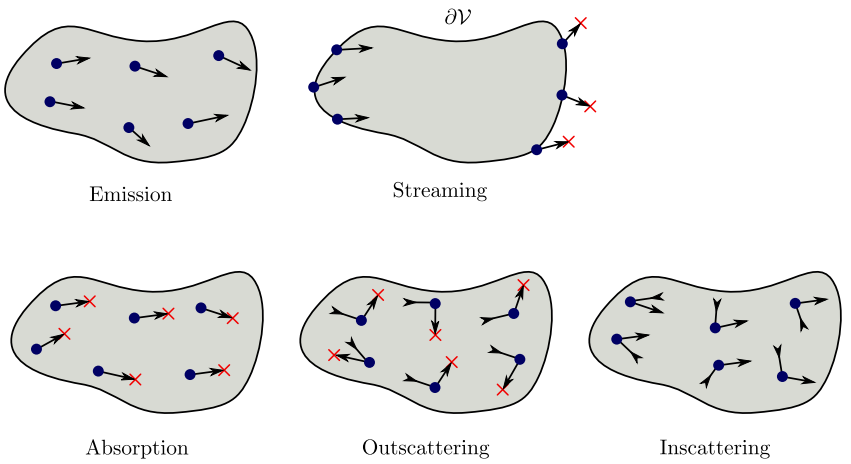
$$E \equiv \int_{\Omega} \int_{\mathcal{V}_0} q_{\mathrm{v}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}) \, \mathrm{d}V(\mathbf{x}_{\mathrm{v}}) \, \mathrm{d}\sigma(\boldsymbol{\omega}).$$

Since E is the change in the number of particles in $\Omega \times \mathcal{V}$ over time, it has units of $1/\mathrm{s}$.

Streaming

Streaming is the change in particle count in $\Omega \times \mathcal{V}$ due to particles entering or leaving through the surface, $\partial\mathcal{V}$, something also referred to as *leakage*. The amount leakage depends on the component of the flux normal to $\partial\mathcal{V}$, so the

Figure 2.4: The five causes of gains and losses of particles in $\Omega \times \mathcal{V}$ shown in Figure 2.3. Emission results in the creation of new particles. Particles enter or leave $\mathcal{V}$ through the boundary $\partial\mathcal{V}$ by streaming. Absorption reduces the number of particles by collisions with the medium. Finally, outscattering causes particles to be scattered out of Ω, and inscattering causes particles to be scattered into Ω.



total change in N due to streaming can be computed as

$$S \equiv \int_{\Omega} \int_{\partial\mathcal{V}} \phi(\mathbf{x}_s, \boldsymbol{\omega})\, \boldsymbol{\omega} \cdot \mathbf{n}(\mathbf{x}_s)\, \mathrm{d}A(\mathbf{x}_s)\, \mathrm{d}\sigma(\boldsymbol{\omega}),$$

where $\mathbf{n}(\mathbf{x}_s)$ is the outward pointing normal at $\mathbf{x}_s$. This means that if S is positive, more particles are leaving $\mathcal{V}$ than entering, i.e. N decreases.

Absorption

Collisions with the host medium can result in the particle getting absorbed. The probability of a particle getting absorbed is proportional to the distance traveled. The constant of proportionality is called the *absorption coefficient*, $\sigma_a(\mathbf{x}_v)$, and has units $1/\mathrm{m}$. The absorption coefficient is a macroscopic description of the interactions of particles with the medium at a microscopic level. It is generally a function of position, but is independent of direction for isotropic media, which is the case considered here. The total change in the number of particles in $\Omega \times \mathcal{V}$

due to absorption can be computed as

$$C_{\text{abs}} \equiv \int_{\Omega} \int_{\mathcal{V}_0} \sigma_{\text{a}}(\mathbf{x}_{\text{v}})\, \phi(\mathbf{x}_{\text{v}}, \boldsymbol{\omega})\, \mathrm{d}V(\mathbf{x}_{\text{v}})\, \mathrm{d}\sigma(\boldsymbol{\omega}).$$

Scattering

Collisions can also result in scattering, which causes particles to change direction instantaneously. The probability distribution which characterizes the scattering behavior of the medium is called *volume scattering kernel*, k_{v}. It is defined so that $k_{\text{v}}(\mathbf{x}_{\text{v}}, \boldsymbol{\omega} \cdot \boldsymbol{\omega}')\, \mathrm{d}V(\mathbf{x}_{\text{v}})\, \mathrm{d}\sigma(\boldsymbol{\omega}')$ is the probability that a particle at $\mathbf{x}_{\text{v}}$ traveling in direction $\boldsymbol{\omega}$ is scattered into direction $\boldsymbol{\omega}'$. For isotropic media, the volume scattering kernel depends only on the relative directions of $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$.

If a particle in $\Omega \times \mathcal{V}$ is scattered into a new direction, $\boldsymbol{\omega}' \notin \Omega$, it is an example of *outscattering*. The total change in the number of particles due to outscattering can be computed as

$$C_{\text{out}} \equiv \int_{\Omega} \int_{\mathcal{V}_0} \int_{\mathcal{S}^2} k_{\text{v}}(\mathbf{x}_{\text{v}}, \boldsymbol{\omega} \cdot \boldsymbol{\omega}')\, \phi(\mathbf{x}_{\text{v}}, \boldsymbol{\omega})\ \mathrm{d}\sigma(\boldsymbol{\omega}')\, \mathrm{d}V(\mathbf{x}_{\text{v}})\, \mathrm{d}\sigma(\boldsymbol{\omega}). \tag{2.1}$$

Similarly particles in $\mathcal{V}$ but with directions outside Ω can be scattered into new directions $\boldsymbol{\omega}' \in \Omega$. This results in an increase in particles and is known as *inscattering*. The total number of particles gained through inscattering can be computed as

$$C_{\text{in}} \equiv \int_{\Omega} \int_{\mathcal{V}_0} \int_{\mathcal{S}^2} k_{\text{v}}(\mathbf{x}_{\text{v}}, \boldsymbol{\omega}' \cdot \boldsymbol{\omega})\, \phi(\mathbf{x}_{\text{v}}, \boldsymbol{\omega}')\ \mathrm{d}\sigma(\boldsymbol{\omega}')\, \mathrm{d}V(\mathbf{x}_{\text{v}})\, \mathrm{d}\sigma(\boldsymbol{\omega}). \tag{2.2}$$

The Balance Equation

Now that the effects that cause changes in the number of particles in $\Omega \times \mathcal{V}$ have been identified, we can group gains and losses to form a balance equation,

$$S + C_{\text{abs}} + C_{\text{out}} = E + C_{\text{in}}. \tag{2.3}$$

Note that in Equation 2.1 and 2.2 the inner integral is over the complete 4π sphere of directions, which includes Ω. This means that inscattering and outscattering both account for particles already in Ω scattered back into Ω. However, since these particles appear both as gains and loses, they effectively cancel each other out.

Equation 2.3 expresses a condition that must be true for a particular subset $\Omega \times \mathcal{V}$ of the phase space in order for equilibrium to exist. We would like to show that this condition is true not only for arbitrary subsets of the phase space, but also for any point $(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega})$ in the phase space. In order to do this, first note that all terms, except the streaming term, have a double integral over Ω and $\mathcal{V}_0$. Streaming is different in that it has an integral over the boundary of $\mathcal{V}$, $\partial\mathcal{V}$. This surface integral can be transformed into a volume integral using the theorem of Gauss. Basically, the integral of flux streaming through the surface is replaced by an integral of the divergence inside the volume, so

$$S = \int_\Omega \int_{\mathcal{V}_0} \boldsymbol{\omega} \cdot \nabla\phi(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) \, \mathrm{d}V(\mathbf{x}_\mathrm{v}) \, \mathrm{d}\sigma(\boldsymbol{\omega}).$$

The intuition behind this is that an integral over the sources and sinks present in the volume must equal the net flow out of the region.

After transforming the streaming term to a volume integral, all five terms contain the same double integral over Ω and $\mathcal{V}_0$. This means that the balance equation must also hold for the integrand alone, so

$$\begin{aligned}\boldsymbol{\omega} \cdot \nabla\phi(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) + \sigma_\mathrm{a}(\mathbf{x}_\mathrm{v}) \, \phi(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) + \int_{\mathcal{S}^2} k_\mathrm{v}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega} \cdot \boldsymbol{\omega}') \, \phi(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) \, \mathrm{d}\sigma(\boldsymbol{\omega}') \\ = q_\mathrm{v}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) + \int_{\mathcal{S}^2} k_\mathrm{v}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}' \cdot \boldsymbol{\omega}) \, \phi(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}') \, \mathrm{d}\sigma(\boldsymbol{\omega}').\end{aligned}$$

The outscattering term can be simplified, since the flux can be moved out of the inner integral. We define the *scattering coefficient*

$$\sigma_\mathrm{s}(\mathbf{x}_\mathrm{v}) = \int_{\mathcal{S}^2} k_\mathrm{v}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega} \cdot \boldsymbol{\omega}') \, \mathrm{d}\sigma(\boldsymbol{\omega}').$$

The scattering coefficient, like the absorption coefficient, has units of $1/\mathrm{m}$ and describes the macroscopic scattering properties of the medium. It is also a function of position, but not direction for isotropic media.

It is convenient to define the sum of the absorption and scattering coefficient as the *extinction coefficient*,[1]

$$\sigma_\mathrm{t}(\mathbf{x}_\mathrm{v}) = \sigma_\mathrm{a}(\mathbf{x}_\mathrm{v}) + \sigma_\mathrm{s}(\mathbf{x}_\mathrm{v}),$$

which is the probability that a particle will collide with medium per unit distance traveled.

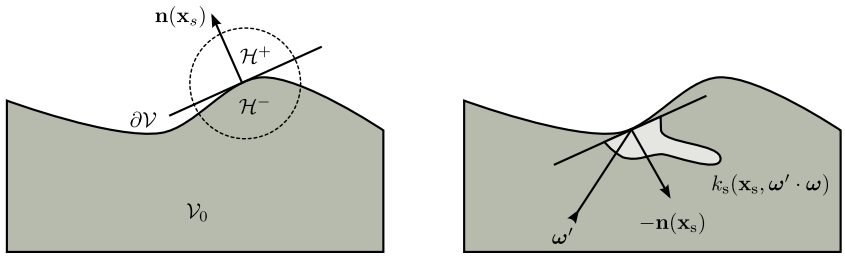
[1] Sometimes also referred to as the total interaction coefficient.

Using the scattering and extinction coefficients, the balance equation can be rewritten as

$$\boldsymbol{\omega} \cdot \nabla\phi(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) + \sigma_\mathrm{t}(\mathbf{x}_\mathrm{v})\,\phi(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) = q_\mathrm{v}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) + \int_{\mathcal{S}^2} k_\mathrm{v}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}' \cdot \boldsymbol{\omega})\,\phi(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}')\;\mathrm{d}\sigma(\boldsymbol{\omega}'), \qquad (2.4)$$

which we will call the equation of transfer.

Figure 2.5: Implicit boundary conditions cause particles that hit the boundary to be reflected back into the volume.



2.2.2 Boundary Conditions

Equation 2.4 is a first order differential equation and needs boundary conditions to fix the otherwise arbitrary constant of integration. Explicit boundary condition are given by

$$\phi(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}) = q_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}),$$

where $q_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega})$ is the surface emission term, measured in $1/\mathrm{m}^2\mathrm{sr}$, which is analogous to the volume emission term. Implicit, or *reflecting*, boundary conditions are given by

$$\phi(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}) = \int_{\mathcal{H}^2_-} k_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}', \boldsymbol{\omega})\,\phi(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}')\;\mathrm{d}\sigma(\boldsymbol{\omega}'),$$

where $k_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}', \boldsymbol{\omega})$ is the *surface scattering kernel*, measured in $1/\mathrm{sr}$, and analogous to the volume scattering kernel. As shown in Figure 2.5, the integral is over the negative hemisphere with respect to the normal $\mathbf{n}(\mathbf{x}_\mathrm{s})$, i.e. over directions pointing into the volume. Both types of boundary conditions can be combined, which yields the more general boundary condition,

$$\phi(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}) = q_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}) + \int_{\mathcal{H}^2_-} k_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}', \boldsymbol{\omega})\,\phi(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}')\;\mathrm{d}\sigma(\boldsymbol{\omega}').$$

2.3 Radiative Transfer

If transport theory is applied to transfer of radiant energy, the field of radiative transfer emerges. In this case, the particles can be thought of as photons, traveling with a speed of $c_0 = 299\,792\,458\,\mathrm{m/s}$. This is assuming vacuum, or free space, since the effective speed of light depends on the host medium.

For photons the phase space has to be extended with at least the notion of *frequency*, ν, or equivalently, *wavelength*, λ, which are related by

$$\nu = \frac{c}{\lambda},$$

where c is the speed of light in the appropriate medium. Polarization can also be included in the state of each photon; however, this component is often ignored in computer graphics.

Limitations of the Particle Model

In the derivation of the equation of transfer, a number of assumptions were made about the behavior of particles in the abstract in order to simplify the resulting model. As discussed in the following, the consequence of this is that certain kinds of lighting effects cannot be accounted for by transport theory.

The first of these limitations is caused by the use of the particle model itself, where it was assumed that particles travel in straight lines. The consequence of using rectilinear propagation is that diffraction cannot be accounted for.

Also, since particle-particle interactions were ignored in the derivation of the equation of transfer, interference cannot be accounted for by this model. This is justified on the grounds, that since most light sources are incoherent, i.e. the phase of the emitted light changes randomly, the chances of interference are small.[2] This means that light energy passing through a point in a given direction is completely independent from light energy passing through the same point but in a different direction.

The model also excludes the phenomenon of phosphorescence. Phosphorescence is time-delayed emission of light. In order to account for this effect, it would be necessary to allow energy to be absorbed at one time and re-emitted later. However, this would greatly increase the complexity of the model, which is not

[2] The most notable exception from this rule is light produced by a laser, which is strongly coherent, and thus cannot be handled accurately by this model.

justified in the majority of scenes, where phosphorescence plays a small role. Similarly, fluorescence, cannot be accounted for, since wavelengths are assumed to be decoupled.

Radiance

The fundamental quantity of interest in radiative transfer is *radiance*. Radiance is radiant power arriving at or leaving a surface (real or imagined) perpendicularly per unit solid angle per unit area. As such, it is a concept similar to phase space flux, which only measured particles rather than power. Fortunately, finding the power of a particle (photon) with a specified frequency can be done using the *Planck relation*,

$$E = h\nu,$$

where h is Planck's constant (6.626×10^{-34} Js). Thus, radiance can be defined in terms of phase space flux as

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\omega}) &= h\nu\, \phi(\mathbf{x}, \boldsymbol{\omega}) \\ &= c\, h\nu\, n(\mathbf{x}, \boldsymbol{\omega}). \end{aligned}$$

and consequently, radiance has units $\mathrm{W}/\mathrm{m}^2\mathrm{sr}$.

It will be convenient to distinguish between *exitant* and *incident* radiance [Veach, 1997, pg. 83]. Exitant radiance, $L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega})$, is radiance leaving a point, $\mathbf{x}$, in direction $\boldsymbol{\omega}$ and incident radiance, $L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega})$, is radiance arriving at a point, $\mathbf{x}$, from direction $\boldsymbol{\omega}$. In free space (no participating medium) these quantities are related by

$$L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}) = L_\mathrm{o}(\mathbf{x}, -\boldsymbol{\omega}).$$

However, in general, the relationship between incident and exitant radiance is more complicated and depends on the scattering and emission properties at $\mathbf{x}$. In the following we will elide the subscript if either quantity can be used.

The remaining terms from transport theory can also be translated into their radiative transfer equivalents. Volume emission is related to the phase space source as

$$L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) \equiv h\nu\, q_\mathrm{v}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}),$$

and has units $\mathrm{W}/\mathrm{m}^3\mathrm{sr}$. Surface emission can be defined in terms of radiance as

$$L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}) \equiv h\nu\, q_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}),$$

and has units $\mathrm{W}/\mathrm{m}^2\mathrm{sr}$.

The scattering kernels can also be rewritten in terms of their radiative transfer equivalents. The volume scattering kernel is replaced by the *phase function*, f_{p},

$$k_{\mathrm{v}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}' \cdot \boldsymbol{\omega}) = \sigma_{\mathrm{s}}(\mathbf{x}_{\mathrm{v}})\, f_{\mathrm{p}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}' \cdot \boldsymbol{\omega})$$

with units $1/\mathrm{sr}$. The surface scattering kernel is replaced by the *bidirectional reflectance-distribution function* (BRDF), f_{r}, given by

$$k_{\mathrm{s}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}', \boldsymbol{\omega}) = f_{\mathrm{r}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}', \boldsymbol{\omega})\, |\cos\theta|,$$

with units $1/\mathrm{sr}$ and where θ is the angle between $\boldsymbol{\omega}'$ and $\mathbf{n}(\mathbf{x}_{\mathrm{s}})$.

Using these terms, the equation of transfer (Equation 2.4) can be rewritten in the more familiar way as

$$\begin{aligned}\boldsymbol{\omega} \cdot \nabla L_{\mathrm{o}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}) + \sigma_{\mathrm{t}}(\mathbf{x}_{\mathrm{v}})\, L_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}}, -\boldsymbol{\omega}) = \\ L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}) + \sigma_{\mathrm{s}}(\mathbf{x}_{\mathrm{v}}) \int_{\mathcal{S}^2} f_{\mathrm{p}}(\mathbf{x}_{\mathrm{v}}, -\boldsymbol{\omega}' \cdot \boldsymbol{\omega})\, L_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}')\, \mathrm{d}\sigma(\boldsymbol{\omega}'),\end{aligned} \tag{2.5}$$

which is known as the *radiative transfer equation*. The same can be done for the boundary conditions, which yields the equation

$$L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}) = L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}) + \int_{\mathcal{H}^2_-} L_{\mathrm{i}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}')\, f_{\mathrm{r}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}', \boldsymbol{\omega})\, |\cos\theta|\ \mathrm{d}\sigma(\boldsymbol{\omega}'). \tag{2.6}$$

If participating media are not present, i.e. if $\sigma_{\mathrm{s}} = \sigma_{\mathrm{a}} = 0$ and $L_{\mathrm{e},\mathcal{V}_0} = 0$, all the terms in Equation 2.5 vanish and all that remains are the boundary conditions. In this case, Equation 2.6 is known as the *rendering equation*, which was first presented by Kajiya [1986], though in a slightly different form.

2.4 Radiometry

Radiance is an example of a radiometric quantity. Radiometry is the science of measurement of electromagnetic radiation. The standard references for radiometric terms are Nicodemus [1976] and Nicodemus, Richmond, Hsia, Ginsberg, and Limperis [1977]. An in-depth discussion of radiometric concepts can also be found in Glassner [1994].

Photometry is a related field, and is covered in Section 2.5. Photometry, which predates radiometry, is the science of the measurement of light as perceived by the human eye. The inclusion of the human visual system makes photometry very non-linear, and consequently radiometry is better suited for performing calculations on light. Radiometric quantities can then be converted to their photometric equivalents afterward, if desired.

Spectral Radiance

It turns out to be convenient to redefine phase space density as a differential quantity with respect to the frequency of light. In this case, the number of particles in $\mathbf{x} \in \mathbb{R}^3$ moving in directions within a differential solid angle of $\mathrm{d}\sigma(\boldsymbol{\omega})$ around the central direction $\boldsymbol{\omega} \in \mathcal{S}^2$ *and* with frequency $\nu \in \mathrm{d}\nu$ is

$$n(\mathbf{x}, \boldsymbol{\omega}, \nu)\ \mathrm{d}V(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega})\ \mathrm{d}\nu$$

or, equivalently, with wavelength $\lambda \in \mathrm{d}\lambda$ is $n(\mathbf{x}, \boldsymbol{\omega}, \lambda)\ \mathrm{d}V(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega})\ \mathrm{d}\lambda$. The notion of phase space flux can also be extended in a similar fashion.

Using these definitions, *spectral radiance* can be defined as

$$L(\mathbf{x}, \boldsymbol{\omega}, \nu) = h\nu\, \phi(\mathbf{x}, \boldsymbol{\omega}, \nu)$$

with units $\mathrm{W}/\mathrm{m}^2\ \mathrm{sr}\ \mathrm{Hz}$. Similarly, the units of spectral radiance measured with respect to wavelength, $L(\mathbf{x}, \boldsymbol{\omega}, \lambda)$, are $\mathrm{W}/\mathrm{m}^2\ \mathrm{sr}\ \mathrm{nm}$.

Radiance is related to spectral radiance as

$$L(\mathbf{x}, \boldsymbol{\omega}) = \int_0^\infty L(\mathbf{x}, \boldsymbol{\omega}, \nu)\ \mathrm{d}\nu = \int_0^\infty L(\mathbf{x}, \boldsymbol{\omega}, \lambda)\ \mathrm{d}\lambda,$$

i.e., it is simply the integral of spectral radiance over all frequencies / wavelengths. As a consequence, non-spectral radiance is not a useful quantity for physically based rendering, since it includes light energy outside the visible spectrum.

Instead, spectral radiance, typically restricted to the visible spectrum, should be used. As noted above, spectral radiance can be measured per unit frequency or per unit wavelength (other choices include the wave number). As discussed in Nicodemus [1976], the most convenient form tends to be per unit wavelength, with wavelength measured in nanometer (nm), so as not to confuse the units of wavelength with the spatial units of area.

Though radiance measured with respect to wavelength is convenient, it has the disadvantage that the wavelength depends on the medium (in particular, it depends on the *index of refraction* of the medium, $\eta(\lambda)$; see Section 2.6). This means that the wavelength changes as the light ray traverses media with different indices of refraction, unlike frequency, which is preserved. This makes both modeling the scene and performing calculations very inconvenient. I.e., consider modeling a light source embedded in a medium with $\eta(\lambda) = 2$. Normally, the light sources used in physically modeling should emit light in the visible range (roughly 360–830 nm). However, due to the changed index of refraction, the

light source should instead emit light in the narrower interval 180–415 nm, but with twice the power. This is to ensure that the light will be in the visible range when it leaves the medium. In addition, as discussed in Section 2.6.1, the radiance of the ray must be scaled by the relative index of refraction cubed, rather than squared, as it crosses the boundary.

To avoid this, *vacuum wavelength* , which is simply the wavelength assuming $\eta(\lambda) = 1$, can be used instead of regular media-dependent wavelength. Vacuum wavelength, λ_0, like frequency, is preserved across media with different indices of refraction.

Sometimes it is necessary to convert radiance measured with respect to frequency to wavelength [Wyszecki and Stiles, 2000]. In order to do so, first note that $L(\lambda)\,\mathrm{d}\lambda = L(\nu)\,\mathrm{d}\nu$. In addition, we know that $\lambda\nu = c$. Differentiating both sides yields

$$\begin{aligned} \mathrm{d}\lambda &= -\frac{c}{\nu^2}\,\mathrm{d}\nu \quad \text{or} \\ \mathrm{d}\nu &= -\frac{c}{\lambda^2}\,\mathrm{d}\lambda, \end{aligned}$$

which can be used directly to convert between different units. The negative sign appears as wavelength and frequency are inversely proportional. The remaining radiometric quantities, which will be described shortly, can be converted in a similar manner. Note that for quantities where the wavelength dependence is purely functional, as opposed to derivative, such as the index of refraction, the relation is much simpler (e.g., $\eta(\lambda) = \eta(\nu)$).

Several other useful radiometric quantities exist, that can be derived from radiance. They exist in both spectral and non-spectral versions; as mentioned, only the spectral versions are really useful, but to simplify notation the wavelength index is omitted in the following.

Radiant Energy and Flux

Radiant flux or *radiant power*, Φ, measured in watts (W), is defined as

$$\Phi = \int_{\Omega}\int_{D_2} L(\mathbf{x}, \boldsymbol{\omega})\,\cos\theta\;\mathrm{d}A(\mathbf{x})\,\mathrm{d}\sigma(\boldsymbol{\omega}).$$

where $D_2 \subset \partial\mathcal{V}$. It is the time derivative of *radiant energy*, Q, which is measured in joules (J). Flux is a useful quantity for measuring the total power of a light source or the total power a sensor receives.

Radiant Intensity

Radiant intensity, with units W/sr, is used to describe the distribution of flux emanating from a point in an infinitesimal beam, with solid angle $\mathrm{d}\sigma(\boldsymbol{\omega})$, in direction $\boldsymbol{\omega}$,

$$I(\boldsymbol{\omega}) = \frac{\mathrm{d}\Phi(\boldsymbol{\omega})}{\mathrm{d}\sigma(\boldsymbol{\omega})}.$$

Radiant intensity is mostly useful for describing the emission of light from point light sources. For instance, an isotropic point light source with power Φ would have a radiant intensity of $\frac{\Phi}{4\pi}$.

Radiant Flux Density

Total flux across a surface measured per unit area is called *radiant flux density*. It is customary to distinguish between incident flux and outgoing, or reflected flux. In the former case, the quantity is referred to as *irradiance* (W/m^2)

$$E(\mathbf{x}) = \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}A(\mathbf{x})}.$$

In the latter, the amount flux leaving a surface per unit area is known as *radiant exitance* (also W/m^2)

$$M(\mathbf{x}) = \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}A(\mathbf{x})}.$$

Irradiance, and its photometric cousin illuminance, can be used to construct iso-lux diagrams. Radiant exitance, which is also known as *radiosity*, is often used to describe the emission from area light sources.

Properties of Radiance

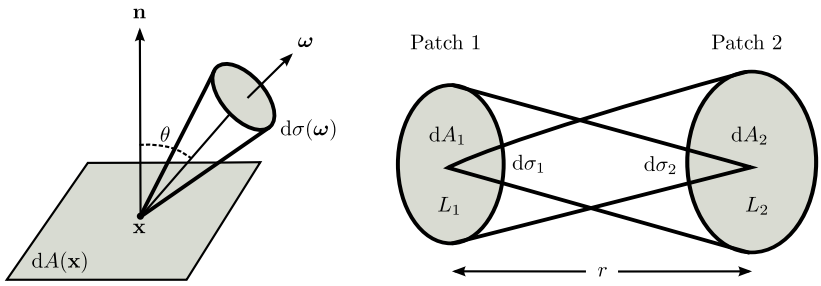
Radiance, presented in the previous section, is in some sense the most fundamental quantity in radiometry, since all other quantities can be derived from it, e.g.

$$L(\mathbf{x}, \boldsymbol{\omega}) = \frac{\mathrm{d}^2\,\Phi(\mathbf{x}, \boldsymbol{\omega})}{\mathrm{d}\sigma(\boldsymbol{\omega})\;\mathrm{d}A(\mathbf{x})\cos\theta} = \frac{\mathrm{d}I(\boldsymbol{\omega})}{\mathrm{d}A(\mathbf{x})\cos\theta} = \frac{\mathrm{d}E(\mathbf{x})}{\mathrm{d}\sigma(\boldsymbol{\omega})\cos\theta}.$$

Radiance is defined as flow across a surface element $\mathrm{d}A$ perpendicularly to the direction $\boldsymbol{\omega}$. If this is not the case, as in Figure 2.6, it is necessary to use the

projected area, $\mathrm{d}A\cos\theta$, instead, which is the reason for the appearance of the cosine term in the denominator. Alternatively, the projected solid angle can be used.

Figure 2.6: The geometry used in the definition of radiance (left). Illustration of the invariance of radiance along a ray between two patches (right).



As discussed in Nicodemus [1963], radiance has the important property of being invariant as it propagates along a ray (assuming vacuum). This makes radiance the right radiometric quantity to associate with a ray and the fundamental quantity to be used in realistic image synthesis.

To make this more concrete, consider Figure 2.6. The total flux leaving patch 1 with differential area $\mathrm{d}A_1$ within the solid angle $\mathrm{d}\sigma_1$ must equal the flux received by patch 2,

$$L_1\ \mathrm{d}\sigma_1\ \mathrm{d}A_1 = L_2\ \mathrm{d}\sigma_2\ \mathrm{d}A_2 \tag{2.7}$$

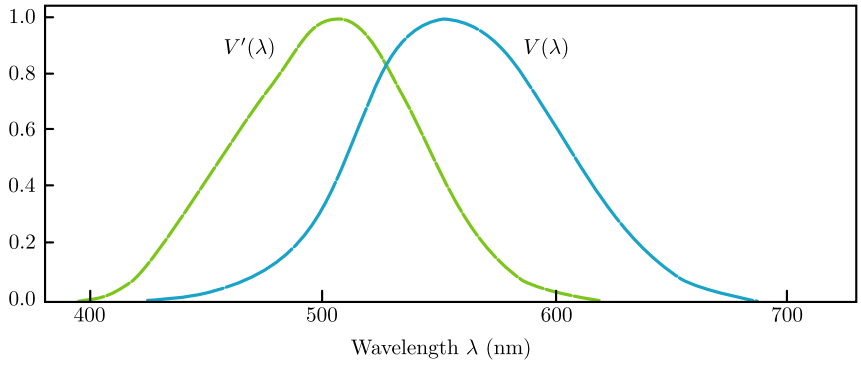
We know from the definition of solid angle that $\mathrm{d}\sigma_1 = \mathrm{d}A_2/r^2$ and $\mathrm{d}\sigma_2 = \mathrm{d}A_1/r^2$. If these expressions are inserted into Equation 2.7, it becomes clear that $L_1 = L_2$, i.e. the radiance leaving the first patch is equal to the incident radiance at the second. This shows that radiance is invariant along a ray.

2.5 Photometry and Color

Spectral radiometric quantities, such as spectral radiance, are useful for performing lighting calculations. However, the result of these calculations are often intended for human viewers and humans cannot sense spectral radiance directly. This means that once we have solved for the equilibrium distribution of spectral radiance, as described by the equation of transfer, it will be necessary to transform the result to a form more suitable for human viewers.

Photometry is the science of measurement of light as perceived by humans. The sensitivity of the human visual system to light varies with wavelength in such a way that light outside the range 360–830 nm is invisible to humans. The function that describes this has been standardized and is known as the spectral luminous efficiency function, $V(\lambda)$. As shown in Figure 2.7, two versions of this function exist; one for photopic vision, which is used when the eye is adapted to light conditions and which is the standard in photometry and one for dark conditions (scotopic vision).

Figure 2.7: The spectral luminous efficiency function for photopic and scotopic vision. The curves are normalized and peak at 555 nm and 507 nm, respectively.



Photometry introduces a host of new quantities, each one corresponding to a radiometric quantity. Only the quantities relevant to this thesis are covered here; see Wyszecki and Stiles [2000] for a complete list.

Luminance is the photometric equivalent of radiance. It can be computed from spectral radiance as[3]

$$L_{\mathrm{v}}(\mathbf{x}, \boldsymbol{\omega}) = K_m \int_0^\infty L(\mathbf{x}, \boldsymbol{\omega}, \lambda)\, V(\lambda)\, \mathrm{d}\lambda,$$

and has units of $^{\mathrm{cd}}/_{\mathrm{m}^2}$ ($^{\mathrm{lm}}/_{\mathrm{sr\ m}^2}$). The constant K_m is 683 $^{\mathrm{lm}}/_{\mathrm{W}}$ for photopic vision. As an example, the luminance of the sun is approximately 1.6×10^9 $^{\mathrm{cd}}/_{\mathrm{m}^2}$ at noon. The photometric equivalent of irradiance is illuminance, which can be computed from spectral irradiance as

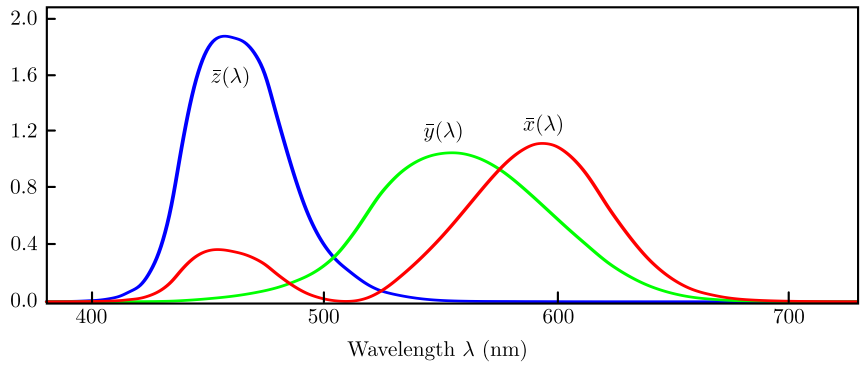
$$E_{\mathrm{v}}(\mathbf{x}) = K_m \int_0^\infty E(\mathbf{x}, \lambda)\, V(\lambda)\, \mathrm{d}\lambda,$$

[3] In practice these integrals can be restricted to the support of $V(\lambda)$, i.e. 360–830 nm.

and has units lux ($^{\mathrm{lm}}/_{\mathrm{m}^2}$). Illuminance is useful for describing how brightly a surface is illuminated. For instance, an appropriate illumination level in an office environment is 100–1000 lux. The illuminance of direct sunlight at equator is roughly 10^5 lux, since the solid angle of the sun is 6.75×10^{-5} sr.

Often the desired result of radiometric calculations is a color image. The human eye has three types of cone cells, each type sensitive to light corresponding roughly to red, green, and blue wavelengths. A consequence of this is that the sensation of color can be described using three parameters, so-called tristimulus values. Based on measurements of the response of the cone cells, a standard observer has been proposed by the International Commission on Illumination (CIE) in 1931. This CIE 1931 2° Standard Observer, characterized by the color matching functions shown in Figure 2.8, defines the CIE 1931 XYZ Color Space.

Figure 2.8: Color matching functions for the CIE 1931 2° Standard Observer. The green color matching function, $\bar{y}(\lambda)$, is identical to spectral luminous efficiency function, $V(\lambda)$.



Converting spectral radiometric quantities is similar to converting to photometric units. In fact, the color matching function corresponding to greenish wavelengths is defined to be identical to the spectral luminous efficiency function. For example, the CIE XYZ coordinates of spectral radiance is

$$X = \int_0^\infty L(\lambda)\,\bar{x}(\lambda)\,\mathrm{d}\lambda, \qquad Y = \int_0^\infty L(\lambda)\,\bar{y}(\lambda)\,\mathrm{d}\lambda, \qquad Z = \int_0^\infty L(\lambda)\,\bar{z}(\lambda)\,\mathrm{d}\lambda.$$

The CIE 1931 XYZ Color Space is designed so that any color can be represented with non-negative weights. Before viewing a color, the CIE XYZ coordinates must be converted into a color space suitable for display, such as sRGB or the color space of a particular monitor. These color spaces are always smaller than

CIE XYZ, i.e. the range of colors they allow is a subset of the colors of CIE XYZ. This means that a choice must be made of how to transform the colors between color spaces, a process called *gamut mapping*. Another issue is that the dynamic range of the display device may be smaller than the dynamic range of the image. This means the luminance values must be compressed in a process called *tone mapping*.

2.6 Scene Description

So far, the equations governing the equilibrium distribution of radiance in a single volume with boundaries and with a description of the scattering properties and light emission have been described. In this section that description will be generalized to multiple volumes, and the notion of *sensors* will be introduced, so that a complete mathematical description of real scene can be made.

We will define a scene as a description of the scattering properties, light sources, and sensors, associated with a finite volume $\mathcal{V} \subset \mathbb{R}^3$. This volume, $\mathcal{V}$, is divided into a number of cells defined by their boundary $\partial\mathcal{V}$ and with interior $\mathcal{V}_0 = \mathcal{V} \setminus \partial\mathcal{V}$. We will assume that it is possible to determine what cell a point $\mathbf{x} \in \mathcal{V}_0$ belongs to or which cells are on either side of a boundary, if $\mathbf{x} \in \partial\mathcal{V}$. When modeling the scene, it is convenient to allow cells to overlap, since this makes modeling certain features easier. However, it should still be possible to unambiguously decide which cell a given point $\mathbf{x} \in \mathcal{V}_0$ belongs to even in case of overlap. A simple way of doing this is by assigning a precedence value to each cell using the technique described by Schmidt and Budge [2002]. scene description

Each cell is assumed to have constant *index of refraction*, $n(\lambda)$, given by the properties of the bulk matter or medium of the cell. The index of refraction is generally wavelength dependent, but is assumed to be constant with respect to position and direction throughout a single cell, and thus piecewise constant throughout the scene. This means that light only refracts (bends) at interfaces between different media, i.e. at cell boundaries. This assumption makes it impossible to account for effects caused by continuously varying index of refraction, such as mirages, but greatly simplifies computations, since light can be assumed to travel in straight lines. This assumption was also made in the derivation of the equation of transfer.

The direction of the refracted ray depends on the relative index of refraction of the two media, and can be found using Snell's Law, which is covered in virtually any book on optics (e.g. Hecht [2002] or Pedrotti, Pedrotti, and Pedrotti [2007]).

Since the refractive index is wavelength dependent, the direction of the refracted ray also depends on wavelength, which is the cause of *dispersion*. A common way to reduce the time needed to solve the light transport problem is to compute the solution for multiple wavelength simultaneously. Dispersion complicates this, so a common approach is to ignore dispersion and use the refractive index at some specific wavelength, typically at the Fraunhofer D line (589.29 nm), in which case the refractive index is denoted n_D, for all wavelengths.

In the general case, the refractive index is a complex number,

$$n(\lambda) = \eta(\lambda) + \mathrm{i}\,\kappa(\lambda),$$

where $\kappa(\lambda)$ is called the attenuation index. Confusingly, the imaginary part of the refractive index is also sometimes called the extinction coefficient, but in this work we will reserve that name for the sum of the absorption and scattering coefficient, σ_t. In fact, the attenuation index turns out to be closely related to the absorption coefficient,

$$\sigma_\mathrm{a}(\lambda) = \frac{4\pi\,\kappa(\lambda)}{\lambda}.$$

The attenuation index can be used to classify materials according to whether they are *dielectrics* or *conductors*. Dielectrics have zero or negligible attenuation index, and thus absorb little or no light. Conductors, on the other hand, cause the electric wave of the light to extinguish, and thus absorb most light. E.g. gold, which is a great conductor, has a refractive index of $(0.34 + 2.77\mathrm{i})$ at 555 nm. This means that a ray of light entering an object made from gold will be reduced to one percent of its original intensity after traveling only some 73 nm. As a consequence, since in computer graphics we model objects on a scale much larger than that of the wavelength of visible light, conductors can for all practical purposes be treated as being completely opaque.

The surfaces that form the boundary between cells are assumed to be two manifolds. Each point on these surfaces, $\mathbf{x}_\mathrm{s} \in \partial\mathcal{V}$, has an associated normal, $\mathbf{n}(\mathbf{x}_\mathrm{s})$. In order to describe the scattering properties of each point, $\mathbf{x}_\mathrm{s}$, first recall that the bidirectional reflectance-distribution function (BRDF), $f_\mathrm{r}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}', \boldsymbol{\omega})$, was introduced to describe reflecting boundary conditions (see Section 2.3). Reflection is scattering of light into the same hemisphere from where it originated. To allow for flow of light energy between cells, it must also be possible for light to scatter into the opposite hemisphere, a process known as transmission. To describe the transmissive scattering properties at a point, the bidirectional transmittance distribution function (BTDF), $f_\mathrm{t}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}', \boldsymbol{\omega})$, can be used in addition to the BRDF.

It turns out to be convenient to introduce the bidirectional scattering distribution function (BSDF), $f_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}', \boldsymbol{\omega})$, to describe both reflection and transmission.

Essentially, a BSDF is formed by two pairs of BRDFs / BTDFs, i.e. by a BRDF and a BTDF for each side of the surface. The BSDF concept does not appear in most radiometry literature and the term seems to have been coined by computer graphics researchers [Veach, 1997, pg. 86]. Using the BSDF, the boundary conditions can be the rewritten to allow for light transfer between cells by replacing the BRDF and changing domain of integration from the negative hemisphere, $\mathcal{H}^2_-$, to the complete sphere of directions, $\mathcal{S}^2$,

$$L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}) = L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}) + \int_{\mathcal{S}^2} L_{\mathrm{i}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}')\, f_{\mathrm{s}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}', \boldsymbol{\omega})\, |\cos\theta|\ \mathrm{d}\sigma(\boldsymbol{\omega}'). \quad (2.8)$$

The scattering properties associated with points in the volume have already been presented in an earlier section, but are repeated here for completeness. Recall that each point in the volume, $\mathbf{x}_{\mathrm{v}} \in \mathcal{V}_0$, has three associated scattering properties. These are the absorption coefficient, $\sigma_{\mathrm{a}}(\mathbf{x}_{\mathrm{v}})$, the scattering coefficient, $\sigma_{\mathrm{s}}(\mathbf{x}_{\mathrm{v}})$, and the phase function, $f_{\mathrm{p}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}' \cdot \boldsymbol{\omega})$. Similarly, the light emission properties of the scene were described earlier, but are also repeated here for completeness. Recall that volume emission is described by the volume emission function, $L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega})$, and that surface emission is described by the surface emission function, $L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega})$.

Table 2.1: The properties necessary for a description of a scene suitable for the light transport problem as presented in this chapter.

Cells ($\mathcal{V}$)	Refractive index	$\eta(\lambda)$
	Attenuation index	$\kappa(\lambda)$
Boundary of cells ($\partial\mathcal{V}$)	BSDF	$f_{\mathrm{s}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}', \boldsymbol{\omega}, \lambda)$
	Surface emission	$L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}, \lambda)$
Interior of cells ($\mathcal{V}_0$)	Phase function	$f_{\mathrm{p}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}' \cdot \boldsymbol{\omega}, \lambda)$
	Absorption coefficient	$\sigma_{\mathrm{a}}(\mathbf{x}_{\mathrm{v}}, \lambda)$
	Scattering coefficient	$\sigma_{\mathrm{s}}(\mathbf{x}_{\mathrm{v}}, \lambda)$
	Volume emission	$L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}, \lambda)$
Sensors ($\mathcal{V}$)	Flux responsivity functions	$W_{\mathrm{e}}^j(\mathbf{x}, \boldsymbol{\omega}, \lambda)$

The last part of the scene description is the specification of the sensors. As discussed in the introduction to this chapter, the light transport problem consists of performing a number of measurements of the equilibrium radiance distribution. Each measurement is the result of the response of a hypothetical sensor to the radiance in the scene. These sensors are defined by their flux responsivity functions, $W_{\mathrm{e}}^j(\mathbf{x}, \boldsymbol{\omega})$. Nicodemus [1978, pg. 58] calls these function R_Φ,

but we will use W_{e}^{j} to mimic the notation used for describing emission. This convention is common in computer graphics, since sensors can be seen as emitting "importance," as discussed further in Chapter 4. The flux responsivity functions are measured in $^{\mathrm{S}}/_{\mathrm{W}}$, where the uppercase S denotes the unit of sensor response, so as not to confuse it with the units of time. Note that unlike radiance, the dependence of W_{e} on position and direction (and wavelength) is purely functional.

All the properties necessary for describing a scene are summarized in Table 2.1.

2.6.1 Surface Scattering

Scattering at surfaces is defined by the BSDF, which in turn is defined in terms of BRDFs and BTDFs. The primary reference for the BRDF is the work of Nicodemus et al. [1977], where the BRDF is derived from the more general bidirectional scattering-surface reflectance-distribution function (BSSRDF).

As illustrated in Figure 2.9, the BSSRDF, denoted $S(\mathbf{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}, \mathbf{x}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{o}})$, is a function that relates differential reflected radiance at one point to differential incident flux at another point,

$$\begin{aligned} \mathrm{d}L_{\mathrm{o}}(\mathbf{x}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{o}}) &= S(\mathbf{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}, \mathbf{x}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{o}})\ \mathrm{d}\Phi_{\mathrm{i}}(\mathbf{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}) \\ &= S(\mathbf{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}, \mathbf{x}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{o}})\, L_{\mathrm{i}}(\mathbf{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}})\, \cos\theta_{\mathrm{i}}\ \mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{i}})\ \mathrm{d}A(\mathbf{x}_{\mathrm{i}}). \end{aligned}$$

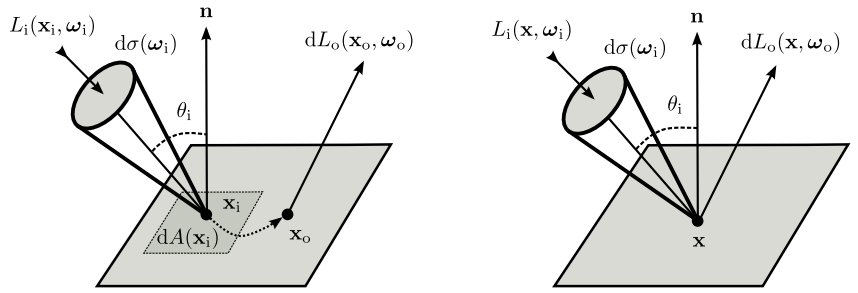
As such, it accounts for light being scattered inside the surface and exiting elsewhere, a phenomenon known as *subsurface scattering*. Unfortunately the eight-dimensional nature of the BSSRDF makes it somewhat impractical to work with. If instead light is restricted exit at the same point it entered the material, the simpler BRDF concept can be used.

Properties of the BRDF

The bidirectional reflectance-distribution function is a four-dimensional[4] function used for describing how light is reflected at a surface. Isotropic BRDFs are a special case, where only the relative orientation of the incident and outgoing directions matters. If the directions are written as spherical coordinates, $\boldsymbol{\omega} = (\theta, \phi)$, then an isotropic BRDF can be written $f_{\mathrm{r}}(\theta_{\mathrm{i}}, \theta_{\mathrm{o}}, \phi_{\mathrm{i}} - \phi_{\mathrm{o}})$, and is consequently a three-dimensional function. For the more general anisotropic BRDF, $f_{\mathrm{r}}(\theta_{\mathrm{i}}, \phi_{\mathrm{i}}, \theta_{\mathrm{o}}, \phi_{\mathrm{o}})$, the absolute directions of the incident and outgoing directions

[4] Six-dimensional, if the spatial parameters are also considered.

Figure 2.9: The BSSRDF (left) describes how differential incident flux at $\mathbf{x}_\mathrm{i}$ is scattered into differential reflected radiance at $\mathbf{x}_\mathrm{o}$ through subsurface scattering. The BRDF (right) describes how differential irradiance is scattered into differential radiance at a single point.



matter. This is necessary to model materials that have oriented patterns in their micro-structure, such as velvet and brushed metals.

Like the BSSRDF, the BRDF is a distribution function, meaning that it can contain generalized functions, such as Dirac delta functions. Mathematically, the BRDF describes the relationship between differential irradiance incident at a point and the differential outgoing radiance at the same point,

$$f_\mathrm{r}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = \frac{\mathrm{d}L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})}{\mathrm{d}E(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})} = \frac{\mathrm{d}L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})}{L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\ \cos\theta_\mathrm{i}\ \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})},$$

as illustrated in Figure 2.9. The BRDF has units sr^{-1} and range $f_\mathrm{r} : \mathcal{H}^2_+ \times \mathcal{H}^2_+ \mapsto [0, \infty)$. To simplify notation, we drop the spatial parameter, $\mathbf{x}$, in the following.

Physically plausible BRDFs have two important properties. Firstly, they are symmetric functions with respect to $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$,

$$f_\mathrm{r}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = f_\mathrm{r}(\boldsymbol{\omega}_\mathrm{o}, \boldsymbol{\omega}_\mathrm{i}),$$

a principle known as Helmholtz reciprocity. Secondly, they conserve energy. The amount of energy a given BRDF reflects is given by the hemispherical-directional reflectance,

$$\rho_{hd}(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathcal{H}^2_+} f_\mathrm{r}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\ \cos\theta_\mathrm{i}\ \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}).$$

To ensure energy conversation, the hemispherical-directional reflectance must obey

$$\rho_{hd}(\boldsymbol{\omega}_\mathrm{o}) \leq 1 \qquad \text{for all } \boldsymbol{\omega}_\mathrm{o} \in \mathcal{H}^2_+.$$

Strictly speaking, $\rho_{hd}(\boldsymbol{\omega}_\mathrm{o}) < 1$, since real surfaces always reflect less that 100%. It is also possible to compute the total fraction of reflected light from any direction,

$$\rho_{hh} = \frac{1}{\pi} \int_{\mathcal{H}^2_+} \int_{\mathcal{H}^2_+} f_\mathrm{r}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) \cos\theta_\mathrm{i} \cos\theta_\mathrm{o} \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{o}),$$

which is known as the hemispherical-hemispherical reflectance.

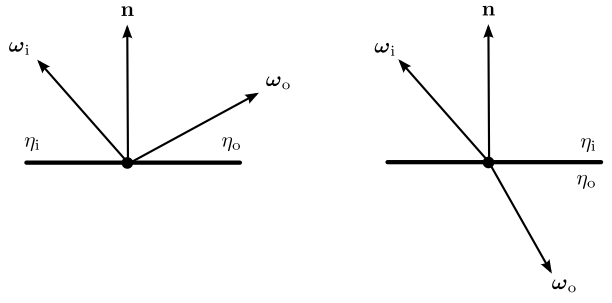
Properties of the BSDF

Veach [1996] shows that any physically valid BSDF must satisfy

$$\frac{f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})}{\eta_\mathrm{i}^2} = \frac{f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i})}{\eta_\mathrm{o}^2}, \tag{2.9}$$

where the arrow notation is used to indicate the direction of light flow (the terms are given in Figure 2.10). This is a generalization of the Helmholtz reciprocity principle for BRDFs, since for reflection $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$ are in the same medium, and thus have the same index of refraction. The implication for transmission is that BTDFs are not symmetric.

Figure 2.10: Scattering by a BSDF with incident direction $\boldsymbol{\omega}_\mathrm{i}$ and outgoing $\boldsymbol{\omega}_\mathrm{o}$. In the case of reflection (left), the BSDF must obey the Helmholtz reciprocity principle, $f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i})$, a special case of Equation 2.9, since $\eta_\mathrm{i} = \eta_\mathrm{o}$. In the case of transmission (right), scattering is between media with different indices of refraction, so $f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = \eta_\mathrm{i}^2/\eta_\mathrm{o}^2 \, f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i})$.



The reason for the appearance of the indices of refraction in Equation 2.9 is that the BSDFs is defined in term of radiance. However, radiance is not invariant across boundaries between media of different refractive indices. *Basic radiance*,

L/η^2, is invariant along a ray[5] even as it crosses smooth boundaries between media with different indices of refraction, assuming no loss to scattering or absorption [Nicodemus, 1963]. It is possible to rephrase the light transport problem in terms of basic radiance, rather than radiance [Veach, 1997, chp. 7], in which case the BSDFs become symmetric even for transmission. However, there are other reasons why the BSDFs used in computer graphics are non-symmetric, and they would still have to be handled explicitly even in this framework.

As discussed in a later chapter, we solve the light transport problem by constructing random paths that connect the sensors and light sources in a scene. By path we simply mean a series of n vertices, $\mathbf{x}_0\mathbf{x}_1 \ldots \mathbf{x}_{n-1}$, with each $\mathbf{x}_i \in \mathcal{V}$, where the first vertex is on a light source and the last vertex is on a sensor. The intermediate vertices are either in $\mathcal{V}_0$, in which case they have an associated phase function, or are in $\partial\mathcal{V}$, in which case they have an associated BSDF. The amount of radiance that is transferred along any of these paths, can be computed based on the emitted radiance, the relative geometry of the vertices of the path, and the scattering at these vertices (more details will be given in Chapter 4).

Bidirectional methods are a class of solution strategies that construct these paths starting both at the light sources and at the sensors. To create a full path, a light source subpath is connected to a sensor subpath. However, note that for paths starting at light sources, the directional arguments to the BSDF cannot simply be exchanged compared to sensor subpaths. The reason is that if the BSDF is non-symmetric, its value will depend on whether it is part of the light or sensor subpath, and as a result, the amount of radiance transferred along the path will depend on how the path was constructed. This is clearly inconsistent, since no matter how a complete light path is constructed, the amount of radiance that flows along the path should be the same.

To remedy this situation, the concept of the *adjoint* BSDF is introduced by Veach [1997, pg. 93]. The adjoint BSDF is similar to the ordinary BSDF, but with the directional arguments exchanged,

$$f_{\mathrm{s}}^*(\boldsymbol{\omega}_{\mathrm{i}} \rightarrow \boldsymbol{\omega}_{\mathrm{o}}) = f_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{o}} \rightarrow \boldsymbol{\omega}_{\mathrm{i}}),$$

and should be used for paths starting at light sources. This way, we can be sure that our results are consistent, no matter how the paths connecting the sensors and light sources were formed.

Non-symmetric BSDFs are also caused by *shading normals*. The basic idea behind shading normals is to replace the true *geometric* normal, which is the

[5] Assuming spectral radiance measured with respect to frequency. If spectral radiance is measured with respect to wavelength, L/η^3 is invariant and Equation 2.9 becomes $f_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{i}} \rightarrow \boldsymbol{\omega}_{\mathrm{o}})/\eta_{\mathrm{i}}^3 = f_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{o}} \rightarrow \boldsymbol{\omega}_{\mathrm{i}})/\eta_{\mathrm{o}}^3$.

normal we have been using so far, with an arbitrary normal that is used only in the lighting calculations. This is a useful trick, that can be used to alter the appearance of a surface in a rather inexpensive way. The earliest example of shading normals seems to be the use of interpolated vertex normals by Phong [1975] to make polygonal meshes appear smooth. Another early use of shading normals was for *bump mapping*, as described by Blinn [1978]. Here, the geometric normal is procedurally altered based on a bump map to produce a shading normal that gives the surface the desired bumpy appearance. In the following, shading normals will be referred to as $\mathbf{n}_\mathrm{s}$, and geometric normals as $\mathbf{n}_\mathrm{g}$.

As discussed by Veach [1997, pg. 151], the shading normal is best is understood as a parameter to the BSDF. To see how shading normals change the appearance, consider again Equation 2.8,

$$L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) = L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) + \int_{\mathcal{S}^2} L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\, f_\mathrm{s}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\, |\mathbf{n}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{i}|\ \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}), \quad (2.10)$$

where we have written out the cosine term as a dot product, $\cos\theta_\mathrm{i} = \mathbf{n}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{i}$. In order to replace the geometric normal with the shading normal, the original BSDF can be replaced by a modified BSDF,

$$f'_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\, \frac{|\mathbf{n}_\mathrm{s} \cdot \boldsymbol{\omega}_\mathrm{i}|}{|\mathbf{n}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{i}|},$$

which will cause the $|\mathbf{n}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{i}|$ term to cancel and effectively replace $\mathbf{n}_\mathrm{g}$ with $\mathbf{n}_\mathrm{s}$. This substitution is often done implicitly; i.e. the shading normal is simply used as a replacement for the geometric normal everywhere. This turns out to be a bad idea, since it has a number of unfortunate side effects. The first is that even if the original BSDF was symmetric, the modified BSDF is not. Instead, the following adjoint BSDF should be used,

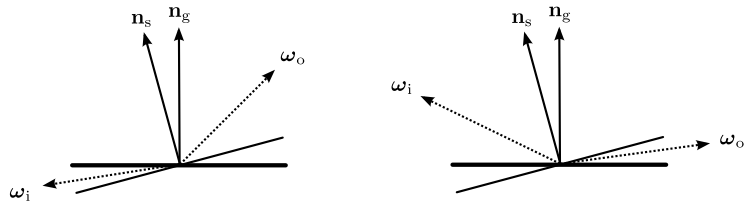
$$f^*_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = f'_\mathrm{s}(\boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i}) = f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i})\, \frac{|\mathbf{n}_\mathrm{s} \cdot \boldsymbol{\omega}_\mathrm{o}|}{|\mathbf{n}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{o}|},$$

whenever paths are created starting from the light sources. Note that this BSDF has an extra factor that depends on the direction we came from ($\boldsymbol{\omega}_\mathrm{o}$) and does not cause any terms to cancel.

A second problem is that shading normals can cause a surface to reflect more energy than it receives. This can happen even if the BSDFs themselves conserve energy. As discussed in Chapter 4, this can cause the equilibrium radiance to become infinite.

As shown in Figure 2.11, shading normals can also cause light leaks and black spots. Light leaks can happen if $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$ lie on different sides of the surface with respect to $\mathbf{n}_\mathrm{g}$, but not with respect to $\mathbf{n}_\mathrm{s}$. This can be fixed by ensuring

Figure 2.11: Shading normals can cause light to leak through surfaces (left). This can be fixed by ensuring that both $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$ are on the same side with respect to the normals. Unfortunately this causes dark spots to appear if $\boldsymbol{\omega}_\mathrm{o}$ fails this test (right).



that $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$ lie on the same side with respect to both normals. However, this results in black spots on the surface, whenever $\boldsymbol{\omega}_\mathrm{o}$ fails this test.

A solution is presented by Veach [1997, pg. 157]. The idea is to extend the BRDF and the BTDF to be full spherical functions, $\mathcal{S}^2 \times \mathcal{S}^2 \mapsto [0; \infty)$. As shown in Algorithm 2.1, the key is to evaluate the BRDF only if the incident and outgoing directions are in the same hemisphere with respect to the geometric normal. Otherwise the BTDF should be evaluated. This approach avoids light leaks, while the extension to $\mathcal{S}^2$ (and the use of absolute values in the dot products) avoids black spots.

Algorithm 2.1: Veach's method for evaluating the BSDF product function for a pair of directions $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$, while taking shading normals into account. This approach prevents both light leaks and black spots.

if $(\boldsymbol{\omega}_\mathrm{i} \cdot \mathbf{n}_\mathrm{g})(\boldsymbol{\omega}_\mathrm{o} \cdot \mathbf{n}_\mathrm{g}) \geq 0$ **then**
  $f = f_\mathrm{r}$
else
  $f = f_\mathrm{t}$
end if
if adjoint **then**
  return $f(\boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i}) |\boldsymbol{\omega}_\mathrm{o} \cdot \mathbf{n}_\mathrm{s}| |\boldsymbol{\omega}_\mathrm{i} \cdot \mathbf{n}_\mathrm{g}| / |\boldsymbol{\omega}_\mathrm{o} \cdot \mathbf{n}_\mathrm{g}|$
else
  return $f(\boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) |\boldsymbol{\omega}_\mathrm{i} \cdot \mathbf{n}_\mathrm{s}|$
end if

In the following, the BSDF and the modified BSDF and its adjoint BSDF for some of the most common scattering functions will be described. These BSDFs are the Lambertian BSDF and the BSDFs for perfect specular reflection and

transmission.

Lambertian Reflection

The Lambertian BRDF, also known as ideal diffuse BRDF [Nicodemus et al., 1977, pg. 43], is given by

$$f_{\mathrm{r,d}}(\boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\rho_d}{\pi},$$

where ρ_d is the reflectance, which is generally a function of both wavelength and position, $\rho_d(\mathbf{x}, \lambda)$. Lambertian surfaces have the property that they appear equally bright regardless of viewing angle, i.e. the reflected radiance from these surfaces is constant.

If shading normals are introduced, the modified Lambertian BRDF can be written

$$f'_{\mathrm{r,d}}(\boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\rho_d}{\pi} \frac{|\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|}{|\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|}, \tag{2.11}$$

and its adjoint can be written

$$f^*_{\mathrm{r,d}}(\boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\rho_d}{\pi} \frac{|\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{o}}|}{|\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{o}}|}. \tag{2.12}$$

When evaluating these functions, it is assumed that $\boldsymbol{\omega}_{\mathrm{i}}$ and $\boldsymbol{\omega}_{\mathrm{o}}$ lie in the same hemisphere with respect to the geometric normal, in the vein of Algorithm 2.1.

Direct application of these BRDFs can cause artifacts in the form of surfaces that appear to be lit from behind, which is especially apparent when using bump mapping. The reason is the use of absolute values in the dot products with the shading normal. Fortunately, the solution is simply to clamp to zero, rather than take absolute value: i.e. use $\max(\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{i}}, 0)$ and $\max(\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{o}}, 0)$ in Equations 2.11 and 2.12 instead. These artifacts do not appear in the results of Veach [1997] (e.g. Figure 5.11a, pg. 163), which implies that clamping was also used in their implementation.

The use of interpolated vertex normals on polygonal meshes can also cause artifacts. As shown in Figure 2.12, the border between light and shadow (which is called the terminator) appears jaggy, and the underlying tessellation of the mesh becomes visible. This artifact appears because the shading normal is used for lighting calculations, while visibility is still determined by the facetted mesh. This inconsistency can result in discontinuities if a point falls in shadow before the cosine between the shading normal and light vector reaches zero.

Figure 2.12: Illustration of the terminator problem on a simple polygonal teapot with increasingly finer tessellation. At the terminator, artifacts appear in the form of discontinuities. This is caused by self-shadowing happening before the cosine-term with the shading normal drops to zero. Increasing the triangle count diminishes the problem, but does not remove it completely.

(a) **(b)** **(c)**

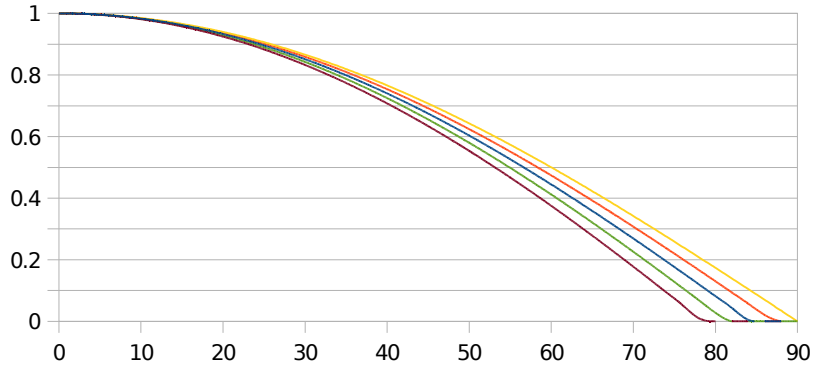
This well known problem is called the *terminator problem* (see Snyder and Barr [1987] and also Woo, Pearce, and Ouellette [1996]). Several partial solutions have been suggested. These include approximating better starting points for shadow rays, which could exacerbate the errors caused by shading normals even further. Higher order surfaces, polygonal meshes with finer tessellation, or displacement maps can also be used, though this can increase the cost of finding intersections considerably. Finally, the use of area light sources, rather than point light sources, and global illumination tend to make the terminator problem less objectionable.

Another solution is presented in Blender Community [2004] (also mentioned in Wächter [2007]). Though not a complete solution, this method is inexpensive and effective in many cases. The idea is to replace the (clamped) dot product in Equations 2.11 and 2.12 with another function,

$$\varphi(\cos\theta, \tau) = \frac{\max(\cos\theta - \tau, 0)}{1 - \tau}.$$

As shown in Figure 2.13, this function is a modified cosine function, where τ is used to control how fast the function drops to zero. This causes the terminator to move further into the illuminated region, eventually causing the artifacts to become hidden in the shadows (see Figure 2.14).

Figure 2.13: The function $\varphi(\cos\theta, \tau)$ for a range of different values of τ $(0.00, 0.05, 0.10, 0.15, 0.20)$. Note how larger values of τ cause the function to drop to zero faster, which has the effect of moving terminator further into the lit region, effectively hiding the aforementioned artifacts.



The optimal value of τ depends on the curvature of the polygonal mesh and the fineness of the tessellation. If τ is chosen too small, artifacts will appear in certain cases and if τ is chosen to large, the object will become unnecessarily dark. Blender Community [2004] suggest computing a single τ value for the entire mesh based on the average dot product of triangle and vertex normals. This can cause problems for meshes that have a highly non-uniform tessellation. In such cases, it would be better to allow τ to vary as a function of position on the mesh.

Using the function φ, the modified Lambertian BRDF can be written

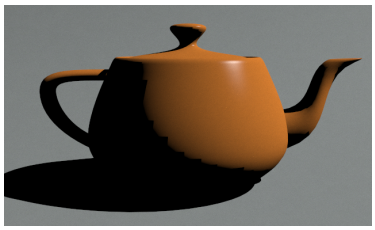
$$f'_{\mathrm{r,d}}(\boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\rho_d}{\pi} \frac{\varphi(\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{i}}, \tau)}{|\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|},$$

and its adjoint can be written

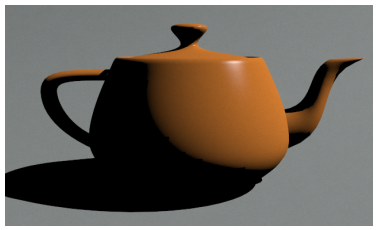
$$f^*_{\mathrm{r,d}}(\boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\rho_d}{\pi} \frac{\varphi(\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{o}}, \tau)}{|\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{o}}|}.$$

Though these function work well in practice, artifacts can still occur if both interpolated vertex normals and bump mapping are used simultaneously.

Figure 2.14: The effect of different values of τ. In the first image, the standard cosine function is used ($\tau = 0$), and artifacts are visible near the terminator. In the next image, a τ value of 0.025 has been used, which has moved the terminator, but some artifacts are still visible. Finally, in the large image, a τ value of 0.1 has been used, which has caused the terminator to move sufficiently to hide all the artifacts.



(a)



(b)

(c)

Perfect Specular Reflection

The BRDF for ideal (lossless) perfect specular reflection, also called mirror-like reflection [Nicodemus et al., 1977], is given by the symmetric function of $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$

$$f_{\mathrm{r,m}}(\boldsymbol{\omega}_\mathrm{i} \rightarrow \boldsymbol{\omega}_\mathrm{o}) = \frac{\delta(\boldsymbol{\omega}_\mathrm{i} - R(\boldsymbol{\omega}_\mathrm{o}, \mathbf{n}))}{|\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{i}|},$$

where the reflection direction is $R(\boldsymbol{\omega}_\mathrm{o}, \mathbf{n}) = 2\,(\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{o})\,\mathbf{n} - \boldsymbol{\omega}_\mathrm{o}$. This BRDF maps a single incident direction to a unique outgoing direction, which is the reason for the appearance of the Dirac delta function.

It is common to combine this BRDF with the Fresnel equations (see e.g. Hecht [2002]), so that the amount of reflected light becomes a function of incident angle and of the indices of refraction. The resulting BRDF can be written

$$f_{\mathrm{r,r}}(\boldsymbol{\omega}_{\mathrm{i}} \rightarrow \boldsymbol{\omega}_{\mathrm{o}}) = F(|\mathbf{n} \cdot \boldsymbol{\omega}_{\mathrm{i}}|) \, \frac{\delta(\boldsymbol{\omega}_{\mathrm{i}} - R(\boldsymbol{\omega}_{\mathrm{o}}, \mathbf{n}))}{|\mathbf{n} \cdot \boldsymbol{\omega}_{\mathrm{i}}|} \tag{2.13}$$

This BRDF can reproduce colored reflections, such as those from metals, since the (possibly complex) indices of refraction are generally wavelength dependent. It also reproduces the effect that surfaces become ideal lossless (colorless) mirrors at grazing angles.

If shading normals are introduced, the modified version of Equation 2.13 becomes,

$$\begin{aligned} f'_{\mathrm{r,r}}(\boldsymbol{\omega}_{\mathrm{i}} \rightarrow \boldsymbol{\omega}_{\mathrm{o}}) &= F(|\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|) \, \frac{\delta(\boldsymbol{\omega}_{\mathrm{i}} - R(\boldsymbol{\omega}_{\mathrm{o}}, \mathbf{n}_{\mathrm{s}}))}{|\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|} \, \frac{|\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|}{|\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|} \\ &= F(|\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|) \, \frac{\delta(\boldsymbol{\omega}_{\mathrm{i}} - R(\boldsymbol{\omega}_{\mathrm{o}}, \mathbf{n}_{\mathrm{s}}))}{|\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{i}}|} \end{aligned} \tag{2.14}$$

where the reflection direction is now found by mirroring around $\mathbf{n}_{\mathrm{s}}$. Note that the denominator in Equation 2.14 still uses the geometric normal, since this term exists to cancel the corresponding term in Equation 2.10. The adjoint BRDF is given by

$$f^{*}_{\mathrm{r,r}}(\boldsymbol{\omega}_{\mathrm{i}} \rightarrow \boldsymbol{\omega}_{\mathrm{o}}) = F(|\mathbf{n}_{\mathrm{s}} \cdot \boldsymbol{\omega}_{\mathrm{o}}|) \, \frac{\delta(\boldsymbol{\omega}_{\mathrm{o}} - R(\boldsymbol{\omega}_{\mathrm{i}}, \mathbf{n}_{\mathrm{s}}))}{|\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{o}}|},$$

which is different from Equation 2.14, since $\mathbf{n}_{\mathrm{s}} \neq \mathbf{n}_{\mathrm{g}}$ implies that $|\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{i}}| \neq |\mathbf{n}_{\mathrm{g}} \cdot \boldsymbol{\omega}_{\mathrm{o}}|$. Again, it is assumed $\boldsymbol{\omega}_{\mathrm{i}}$ and $\boldsymbol{\omega}_{\mathrm{o}}$ lie in the same hemisphere with respect to the geometric normal (otherwise the functions evaluate to zero).

Perfect Specular Transmission

The BTDF for perfect specular transmission [Veach, 1997, pg. 145] is given by

$$f_{\mathrm{r,t}}(\boldsymbol{\omega}_{\mathrm{i}} \rightarrow \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\eta_{\mathrm{o}}^2}{\eta_{\mathrm{i}}^2} \, \frac{\delta(\boldsymbol{\omega}_{\mathrm{i}} - T(\boldsymbol{\omega}_{\mathrm{o}}, \mathbf{n}))}{|\mathbf{n} \cdot \boldsymbol{\omega}_{\mathrm{i}}|}, \tag{2.15}$$

where $T(\boldsymbol{\omega}_{\mathrm{o}}, \mathbf{n})$ is the refracted direction, which can be computed using Snell's Law and the indices of refraction (defined in Figure 2.10). The scaling by the relative index of refraction is necessary to account for the change in radiance that occurs when an interface is crossed. For instance, radiance scattered into a medium with higher index of refraction will be intensified, since the solid angle

of the ray in the dense medium will be smaller. This happens because the full incident hemisphere is compressed into a partial hemisphere, where the missing part is the region where total internal reflection occurs.

The adjoint BTDF for transmission can be derived from Equations 2.9 and 2.15, and is given by

$$f^*_{\mathrm{r,t}}(\boldsymbol{\omega}_\mathrm{i} \rightarrow \boldsymbol{\omega}_\mathrm{o}) = \frac{\eta_\mathrm{i}^2}{\eta_\mathrm{o}^2} f_{\mathrm{r,t}}(\boldsymbol{\omega}_\mathrm{o} \rightarrow \boldsymbol{\omega}_\mathrm{i}) = \frac{\delta(\boldsymbol{\omega}_\mathrm{o} - T(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}))}{|\mathbf{n} \cdot \boldsymbol{\omega}_\mathrm{o}|},$$

where it is seen that term involving the relative indices of refraction cancels. This is sometimes explained as being because the adjoint BTDF is used to propagate flux, which, unlike radiance, is conserved across boundaries between media with different indices of refraction.

If shading normals are introduced, Equation 2.15 becomes

$$\begin{aligned} f'_{\mathrm{r,t}}(\boldsymbol{\omega}_\mathrm{i} \rightarrow \boldsymbol{\omega}_\mathrm{o}) &= \frac{\eta_\mathrm{o}^2}{\eta_\mathrm{i}^2} \frac{\delta(\boldsymbol{\omega}_\mathrm{i} - T(\boldsymbol{\omega}_\mathrm{o}, \mathbf{n}_\mathrm{s}))}{|\mathbf{n}_\mathrm{s} \cdot \boldsymbol{\omega}_\mathrm{i}|} \frac{|\mathbf{n}_\mathrm{s} \cdot \boldsymbol{\omega}_\mathrm{i}|}{|\mathbf{n}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{i}|} \\ &= \frac{\eta_\mathrm{o}^2}{\eta_\mathrm{i}^2} \frac{\delta(\boldsymbol{\omega}_\mathrm{i} - T(\boldsymbol{\omega}_\mathrm{o}, \mathbf{n}_\mathrm{s}))}{|\mathbf{n}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{i}|}. \end{aligned}$$

Similarly, the adjoint BTDF for refraction with shading normals can be written

$$f^*_{\mathrm{r,t}}(\boldsymbol{\omega}_\mathrm{i} \rightarrow \boldsymbol{\omega}_\mathrm{o}) = \frac{\delta(\boldsymbol{\omega}_\mathrm{i} - T(\boldsymbol{\omega}_\mathrm{o}, \mathbf{n}_\mathrm{s}))}{|\mathbf{n}_\mathrm{s} \cdot \boldsymbol{\omega}_\mathrm{i}|} \frac{|\mathbf{n}_\mathrm{s} \cdot \boldsymbol{\omega}_\mathrm{o}|}{|\mathbf{n}_\mathrm{g} \cdot \boldsymbol{\omega}_\mathrm{o}|}$$

Again, it is assumed that the BTDFs evaluates to zero if $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$ are in the same hemisphere with respect to $\mathbf{n}_\mathrm{g}$. Like the BRDF for perfect specular reflection, the BTDF for perfect specular transmission can also be combined with a Fresnel term.

2.6.2 Volume Scattering

Participating media, such as fog and smoke, cause volume scattering to occur. Recall that the scene is modeled as a set of connected cells, where each cell is assigned volume scattering properties, which may depend on position inside the cell, such as would be the case for a heterogeneous media like smoke. We will further assume that the bulk index of refraction of a cell is the same as that of the surrounding cells, such that scattering does not occur on the boundary of the cell. Essentially, this means that the boundary can be ignored with respect to scattering.

If the indices of refraction are different between neighboring cells with participating media, we will assume that a BSDF exists at the boundary, which will cause light to scatter and refract into the cell. If this is the case, subsurface scattering is said to occur, which is the effect responsible for the soft appearance of many materials, such as skin, milk, and marble. Subsurface scattering is usually treated as a different phenomenon than ordinary volume scattering, though the underlying equations are the same.

Recall that the volume scattering properties of a medium are determined by the phase function, and the scattering and absorption coefficients. Specifically, the phase function determines the distribution of scattered radiance at any point in the volume, and thus serves the same function as the BSDF for surfaces.

Like BRDFs, phase functions are Helmholtz reciprocal,

$$f_{\mathrm{p}}(\boldsymbol{\omega}_{\mathrm{i}} \rightarrow \boldsymbol{\omega}_{\mathrm{o}}) = f_{\mathrm{p}}(\boldsymbol{\omega}_{\mathrm{o}} \rightarrow \boldsymbol{\omega}_{\mathrm{i}}).$$

However, unlike the surface scattering case, there are no shading normals to ruin reciprocity and also no asymmetry due to refraction and therefore the concept of an adjoint phase function does not need to be introduced. To make matters even simpler, for isotropic media, which is the case considered here, phase functions are only a function of the relative directions, so $f_{\mathrm{p}}(\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{o}}) = f_{\mathrm{p}}(\cos\theta)$.

Unlike BRDFs, phase functions are normalized, so

$$\int_{\mathcal{S}^2} f_{\mathrm{p}}(\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{o}}) \, \mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{i}}) = 1 \qquad \text{for all } \boldsymbol{\omega}_{\mathrm{o}} \in \mathcal{S}^2.$$

Instead, the amount of scattering is controlled by the scattering coefficient, σ_{s}. The directional arguments to the phase function follow a different convention than that of the BSDF, since the incident direction is reversed (i.e., $\theta = 0$ if $\boldsymbol{\omega}_{\mathrm{i}} = \boldsymbol{\omega}_{\mathrm{o}}$).

The Henyey and Greenstein [1941] phase function is arguably the most commonly used phase function. It is given by

$$f_{\mathrm{p,s}}(\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{o}}) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{\frac{3}{2}}},$$

where $g \in \,]-1;1[$ is called the asymmetry parameter. This parameter controls the degree to which the function is forward scattering (positive g) or backward scattering (negative g). If $g = 0$, the Henyey-Greenstein phase function is reduced to the isotropic phase function,

$$f_{\mathrm{p}}(\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{o}}) = \frac{1}{4\pi}.$$

2.6.3 Light Sources

Light emission happens either as volume emission or surface emission. Emission of light from surfaces is by far the most common situation in computer graphics, though volume emission is also used occasionally (for instance, volume emission would be necessary for accurately modeling a candle flame). Computer graphics light sources also include the ubiquitous point light source, which is an example of a zero dimensional light source, and also linear (1D) light sources.

The classical isotropic point light source can be described using a delta function as

$$L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x},\boldsymbol{\omega}) = \frac{\Phi(\lambda)}{4\pi}\,\delta(\mathbf{x}_{\mathrm{p}} - \mathbf{x}),$$

where $\Phi(\lambda)$ is the spectral power distribution (SPD) and $\mathbf{x}_{\mathrm{p}} \in \mathcal{V}_0$ the position. A spotlight can be modeled by restricting the emission directions of a point light to a cone or other shape [Barzel, 1997].

Light sources can be classified according to whether they are natural or artificial. The most important natural light source is of course the sun. The source of skylight is also the sun, but the light has undergone Rayleigh and Mie scattering. Simulating these phenomena can be time consuming, so in computer graphics the sky is often treated as a separate distant light source where the emission has been precomputed, as has been done in Preetham, Shirley, and Smits [1999] for daylight. Other models for skylight include Haber, Magnor, and Seidel [2005] for twilight, and in Jensen, Durand, Dorsey, Stark, Shirley, and Premože [2001] for the night sky.

Instead of proving a mathematical model, light sources can also be measured. Natural light can be measured using light probes, as is done in Debevec [1998], and used as background illumination. Artificial light sources can also be measured and used as input for rendering algorithms (see e.g. Goesele, Granier, Heidrich, and Seidel [2003]).

Rather than having explicit light source objects, all surfaces in a scene are assumed to have an emission term. The area light sources are then simply the subset of the surfaces that have a nonzero emission term. This means that surfaces that emit light, can also scatter light, since these surfaces also have a BSDF. The simplest form of surface emission is when the emission characteristics only varies as a function of wavelength, such as when emission is described by an ideal blackbody emitter of temperature T (in Kelvins),

$$L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_{\mathrm{s}},\boldsymbol{\omega},\lambda) = \frac{M_{\mathrm{e}}(T,\lambda)}{\pi}.$$

The spectral radiant exitance of blackbody radiator can be computed using a variation of Planck's formula [Wyszecki and Stiles, 2000][6]

$$M_{\mathrm{e}}(T,\lambda) = \frac{c_1 \lambda^{-5}}{\exp\left(\frac{c_2}{T\lambda}\right) - 1},$$

which has units $\mathrm{W}/\mathrm{m}^2\,\mathrm{m}$. The two radiation constants are $c_1 = 2\pi hc^2 = 3.741832 \times 10^{-16}\,\mathrm{W}\cdot\mathrm{m}^2$ and $c_2 = \frac{hc}{k} = 1.438786 \times 10^{-2}\,\mathrm{m}\cdot\mathrm{K}$, where $k = 1.380662 \times 10^{-23}\,\mathrm{J}/\mathrm{K}$ is the Boltzmann constant.

Another option is to use the illuminants defined by International Commission on Illumination (CIE). These include the CIE Illuminant A, which is designed to model light created due to incandescence, such as that from a standard tungsten-filament light bulb. The spectrum for this illuminant can be computed directly using Planck's formula (with $T = 2856\,\mathrm{K}$). Illuminants B and C represent direct sunlight and average daylight (correlated color temperatures of 4874 K and 6774 K, respectively), but have fallen into disuse. CIE Illuminants D are a whole family of relative spectral power distributions for daylight simulation, of which D_{55}, D_{65}, and D_{75} are best known. They can be computed using a simple formula based on a tabulated set of basis functions [Wyszecki and Stiles, 2000] and are used extensively in various fields. E.g. D_{65} is the white point of the sRGB color space mentioned previously. Finally, Illuminant E is an equal energy spectrum, and the Illuminant F series can be used to approximate the spiky SPDs of fluorescent lighting.

If surface emission is constant with respect to $\boldsymbol{\omega}$, the resulting light source is said to be ideal diffuse. A simple empirical formula can be used to create a more directional light source, such as a spotlight. The idea, which is similar to what is proposed in Warn [1983], is to let the falloff away from the normal be controlled by a cosine lobe,

$$L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_{\mathrm{s}},\boldsymbol{\omega},\lambda) = \frac{n+1}{2\pi}(\mathbf{n}_{\mathrm{s}}\cdot\boldsymbol{\omega})^{n-1}\,M_{\mathrm{e}}(T,\lambda)$$

where the exponent $n \geq 1$ controls the speed of the falloff ($n = 1$ gives a diffuse emitter).

The emission of a light source can also be controlled by actually modeling the light fixture. This is the most accurate solution, but can be impractical in some cases. Verbeck and Greenberg [1984] suggest another option, which is to use goniometric diagrams to describe the directional emission properties of light sources. Goniometric diagrams are a standard tool in the lighting industry and are often made available from lamp manufacturers. Surface emission can also

[6] This formula assumes λ is measured in meters, rather than nanometers.

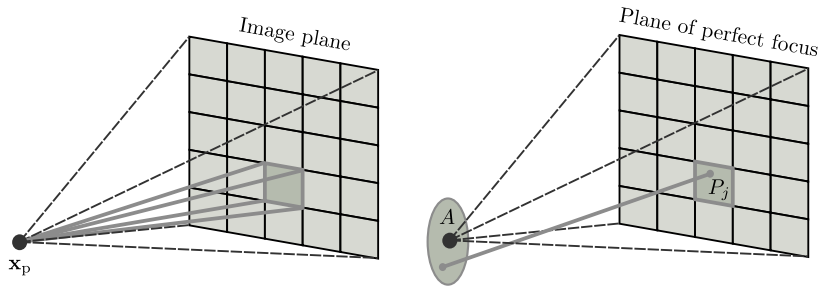
be a function of position. As discussed by Gershbein, Schröder, and Hanrahan [1994], this can be useful to model the emission from a computer monitor, where the radiant exitance would be encoded in a high dynamic range texture map.

2.6.4 Sensors

Solving the light transport problem entails computing how a set of sensors responds to the equilibrium radiance distribution. The response of sensor j is a single real number, called a *measurement*, which is denoted I_j.

Cameras are the most common form of sensor in computer graphics. However, in this terminology, it is more correct to think of cameras as collections of sensors, since real cameras perform multiple measurements simultaneously. This means that the value of each pixel corresponds to a single measurement. E.g., if we are modeling a digital camera, each photosensor, j, on the imaging chip would have its own unique flux responsivity function, W_{e}^j.

Figure 2.15: The flux responsivity function for a single pixel from a pinhole camera (left) is non-zero only for the set of directions within the pyramid formed by the pinhole and the four corners of the corresponding pixel. Similarly, $W_{\mathrm{e}}^j(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega})$ for a camera with a finite aperture (right) is nonzero only for the set of directions formed by connecting points on the aperture to points within the footprint of the pixel in the plane of perfect focus.



The simplest camera model used in computer graphics is the pinhole camera. The flux responsivity of a single pixel of such a camera can be written,

$$W_{\mathrm{e}}^j(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}) = W_{\mathrm{e}}^j(\boldsymbol{\omega})\, \delta(\mathbf{x}_{\mathrm{p}} - \mathbf{x}_{\mathrm{v}}),$$

where $\mathbf{x}_{\mathrm{p}} \in \mathcal{V}_0$ is the pinhole. The function $W_{\mathrm{e}}^j(\boldsymbol{\omega})$ is the image reconstruction filter. It is defined to be nonzero only within the solid angle subtended by the support of the filter function in the image plane as seen from the pinhole (see

Figure 2.15). It is well known that rather than using a constant reconstruction filter, which has a poor frequency response, better results can be achieved using the filter suggested by Cook [1986] (a truncated Gaussian) or the filters described in Mitchell and Netravali [1988] (a family of filters that provide trade-off between ringing and blurring).

The pinhole camera can be modified to account for light incident from the complete 4π sphere of directions. Such a camera is sometimes called a light probe camera, since it can be used to create synthetic light probes. The definition is similar to the pinhole camera, except that function $W_{\mathrm{e}}^{j}(\boldsymbol{\omega})$ is now defined over a full spherical image plane, instead of being based on the rectangular image plane of the pinhole camera. Note the similarity of the definitions of the pinhole camera and light probe camera to the definitions of the spotlight and isotropic point light source in the previous section.

Real cameras have a lens and a finite aperture, which is the cause of depth of field. Such cameras can be modeled using a finite aperture camera model [Potmesil and Chakravarty, 1981]. The aperture, which usually has a circular shape, should be modeled as being a part of the scene, $A \subset \partial\mathcal{V}$ (see Figure 2.15). The flux responsivity function for this camera is non-zero only for the pairs of points and directions, $(\mathbf{x}, \boldsymbol{\omega})$, that satisfy

$$\{(\mathbf{x}, \boldsymbol{\omega}) \mid \mathbf{x} \in A \text{ and } \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}} \text{ where } \mathbf{y} \in P_j\},$$

where P_j is the footprint of the jth pixel in the plane of perfect focus (the distance to the plane of perfect focus is given as input to model for this kind camera). The use of a lens causes light to refract, so that it focuses on the image plane. It is common to approximate the refraction in the lens using a thin lens model, so that only one refraction is accounted for. On the other hand, if the goal is to match the geometry of image formation of a real camera, such a simple model will not suffice. Instead Kolb, Mitchell, and Hanrahan [1995] suggest modeling a camera based on a description of apertures, stops, and lens elements and using the more accurate thick lens model.

So far we have not considered the wavelength dependence of the flux responsivity functions or the camera response, which can be nonlinear. If an ideal camera is assumed, the response is linear, and the CIE color matching functions can be used to compute tristimulus values,

$$W_{\mathrm{e}}^{j}(\mathbf{x}, \boldsymbol{\omega}, \lambda) = W_{\mathrm{e}}^{j}(\mathbf{x}, \boldsymbol{\omega})\, \bar{y}(\lambda),$$

and similarly for $\bar{x}$ and $\bar{z}$. A camera defined this way has a gamut that encompasses all colors, unlike real cameras whose gamut is always a subset of this. If the goal is to match the output of a real camera, Grossberg and Nayar [2003] suggest a method for measuring the camera response. This method

was developed for computer vision, where the opposite problem is faced; i.e., scene radiance needs to be estimated based on the camera response. For computer graphics we need to invert the response function, so that camera response can be computed based on scene radiance. However, we will assume that such adjustments are done in a postprocess, so that all sensors are linear.

2.7 The Measurement Equation

As discussed, linear sensors are defined by their flux responsivity functions, W_{e}^{j}, which define a weighting on the radiance in the scene. Using these functions, the notion of a *measurement* can be defined for each sensor as

$$I_j = \int_0^\infty \int_{\mathcal{S}^2} \int_{\partial\mathcal{V}} L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}, \lambda)\, W_{\mathrm{e}}^{j}(\mathbf{x}, \boldsymbol{\omega}, \lambda)\, |\cos\theta|\ \mathrm{d}A(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega})\ \mathrm{d}\lambda,$$

where the units of I_j are sensor response, S. Consequently, this equation is known as the *measurement equation* [Nicodemus, 1978, pg. 65]. This equation describes measurements across sensors defined as part of $\partial\mathcal{V}$, such as a camera with a finite aperture.

We will extend Nicodemus' definition of a measurement to also include measurements over points from $\mathcal{V}_0$, such as those resulting from the light probe camera mentioned previously. In that case, since there is no surface, and consequently no normal associated with $\mathbf{x}$, the cosine term disappears, and the measurement equation becomes

$$I_j = \int_0^\infty \int_{\mathcal{S}^2} \int_{\mathcal{V}_0} L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}, \lambda)\, W_{\mathrm{e}}^{j}(\mathbf{x}, \boldsymbol{\omega}, \lambda)\ \mathrm{d}V(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega})\ \mathrm{d}\lambda.$$

It will be convenient to combine these two equations, so that we can perform measurements regardless of whether $W_{\mathrm{e}}^{j}(\mathbf{x}, \boldsymbol{\omega})$ is defined to be nonzero across $\partial\mathcal{V}$ or $\mathcal{V}_0$. This can be done using the α measure, defined in Section 2.1, as a measure on $\mathcal{V}$. Using this measure, the measurement equation can be written

$$I_j = \int_0^\infty \int_{\mathcal{S}^2} \int_{\mathcal{V}} L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}, \lambda)\, W_{\mathrm{e}}^{j}(\mathbf{x}, \boldsymbol{\omega}, \lambda)\ \mathrm{d}\alpha_{\boldsymbol{\omega}}(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega})\ \mathrm{d}\lambda. \tag{2.16}$$

The light transport problem is the task of solving the measurement equation for each sensor in the scene. Often it is desirable to perform several measurements, $I_1, \ldots, I_n$, simultaneously, since some computations can be shared. For instance, to construct an image, Equation 2.16 must be solved for each pixel.

The end result need not be an image, however. As discussed in the previous section, the sensor formalism is not restricted to cameras in the classical sense and can easily be extended to more general measurements, such as sensors for measuring irradiance across a surface.

2.8 Summary

In this chapter a complete description of the light transport problem suitable for realistic image synthesis has been presented. This description consists of three parts.

The first part is the description of the scene itself. This includes the geometry that make up the surfaces of the scene. It includes the scattering properties associated with these surfaces and the volumes they enclose. It also includes a description of the emission from light sources and the responsivity of the sensors in the scene.

The second part is the equation of transfer with boundary conditions given by the local scattering equation (rendering equation). These equations describe conditions that must apply to the equilibrium radiance of the scene.

$$\boldsymbol{\omega} \cdot \nabla L_{\mathrm{o}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}, \lambda) + \sigma_{\mathrm{t}}(\mathbf{x}_{\mathrm{v}}, \lambda)\, L_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}}, -\boldsymbol{\omega}, \lambda) = L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}, \lambda) + \sigma_{\mathrm{s}}(\mathbf{x}_{\mathrm{v}}, \lambda) \int_{\mathcal{S}^2} f_{\mathrm{p}}(\mathbf{x}_{\mathrm{v}}, -\boldsymbol{\omega}' \cdot \boldsymbol{\omega}, \lambda)\, L_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}', \lambda)\, \mathrm{d}\sigma(\boldsymbol{\omega}')$$

$$L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}, \lambda) = L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}, \lambda) + \int_{\mathcal{S}^2} L_{\mathrm{i}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}', \lambda)\, f_{\mathrm{s}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}', \boldsymbol{\omega}, \lambda)\, |\cos\theta|\ \mathrm{d}\sigma(\boldsymbol{\omega}')$$

The third and last part is the measurement equation. This equation must be solved for each for each sensor in the scene, leading to a number of measurements, $I_1, \ldots, I_n$, which is the solution to the light transport problem.

$$I_j = \int_0^\infty \int_{\mathcal{S}^2} \int_{\mathcal{V}} L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}, \lambda)\, W_{\mathrm{e}}^j(\mathbf{x}, \boldsymbol{\omega}, \lambda)\ \mathrm{d}\alpha_{\boldsymbol{\omega}}(\mathbf{x})\ \mathrm{d}\sigma(\boldsymbol{\omega})\ \mathrm{d}\lambda$$

We can now turn to algorithms for solving the light transport problem, which is the topic of the next chapters.

Chapter 3

Monte Carlo Methods

Monte Carlo methods are a set of mathematical algorithms developed in the 1940s for solving hard integration problems. They are based on the idea that many interesting problems can be investigated using the outcomes of games of chance. In fact, the name "Monte Carlo" is a reference to the Monte Carlo casino in the city-state of Monaco on the French Riviera, a place famous for gambling.

The use of games of chance, or random sampling, makes the outcome of these algorithms non-deterministic and subject to statistical interpretation. By random sampling we simply mean that the outcome of some number of random events are used as input to the algorithm, and usually, as the number events are increased, the result of the algorithm will converge toward the correct result.

Early methods used "analog" random events, such as rolling a dice or drawing numbered pieces of paper from a bowl. Modern Monte Carlo methods are implemented on computers and use sequences of random or pseudo random numbers.

Monte Carlo methods can be used to solve both probabilistic and deterministic problems. Probabilistic problems are those where the underlying model is of a statistical nature, such as a physical random process. A simple example is the behavior of a photon. Here the scattering direction of the photon can be seen as

a random process that could be directly simulated using random numbers. In contrast, deterministic problems do not have an underlying random structure that can be directly simulated using Monte Carlo. To solve these problems using Monte Carlo methods we must construct a, perhaps unrelated, random process that can be simulated and from which a statistic can be drawn that corresponds to the solution of the original problem. An example of deterministic problems are the integro-differential equations investigated in this thesis. These equations do not correspond to any random process. However, as these equations describe light transport, it is not difficult to come up with a suitable random process, namely the scattering of photons, that can be used to investigate this problem.

3.1 Background

Historically, it is difficult to determine the first use of random sampling to solve scientific problems. There are several known isolated instances, several of which are discussed below. However, none of these people seem to have recognized the use of random sampling as a principled way of solving scientific problems.

An early example, which is often quoted, is that of the Comte de Buffon. In 1777 he determined the probability that a needle of length L thrown at random on piece of paper with straight lines a distance $d > L$ apart would intersect one of the lines. Using analysis he deduced the probability, p, to be

$$p = \frac{2L}{\pi d},$$

and confirmed his result experimentally by performing many throws. Later, Laplace noted that this method could be used to determine π. Either way, this method is essentially an analog Monte Carlo method for estimating p or π.

In 1901 Lord Kelvin appears to have used random sampling to study the kinetic theory of gasses. A few years later, in 1908 Student (W. S. Gosset) used experimental sampling to find an expression for the distribution of the correlation coefficient and later to bolster his confidence in his t-distribution. In 1928 Courant, Friedrichs, and Lewy showed the equivalence of a class of random walks to the solution of certain partial differential equations and in 1931 Kolmogorov proved the relationship between Markov stochastic processes and integro-differential equations.

See Kalos and Whitlock [1986] and Hammersley and Handscomb [1964] for further examples of the early uses of Monte Carlo methods.

The group of people most directly responsible for the "modern" Monte Carlo methods are some of the individuals brought together at Los Alamos in the 1940s. The Los Alamos National Laboratory, part of the top secret Manhattan Project, and known then as Site Y, was founded in 1943 to conduct research on nuclear weapons. This extraordinary group included people such as Richard Feynman, Stanislaw Ulam, Hans Bethe, Enrico Fermi, John von Neumann, Edward Teller, and Nicholas Metropolis.

Unfortunately, early attempts at using Monte Carlo methods at Los Alamos were hampered by the lack of digital computers at that time. It was not until a few years after the war that computers had matured enough to be useful for Monte Carlo. This first happened with the ENIAC, which was used in the development of the "super," the first thermonuclear weapon, and also ran the first computerized Monte Carlo experiments.

In the following years Monte Carlo methods were applied to a number of different problem. These experiences lead to the famous paper "The Monte Carlo Method" by Metropolis and Ulam [1949].

Looking back, Monte Carlo methods have been highly successful and are today used in many fields. There are several reasons for the success of Monte Carlo methods compared to other techniques. The most important are

- Convergence rates are $O(N^{-\frac{1}{2}})$ independent of dimension
- They handle non-smooth functions and functions with singularities
- They are general, and often simple to implement
- They are well suited for implementation on computers

The remainder of chapter is organized as follows: First the necessary probability theory needed for understanding Monte Carlo is reviewed. Then the Monte Carlo method itself is presented in Section 3.3. Finally, the class of methods known as Markov Chain Monte Carlo.

3.2 Probability Theory

Probability plays a central role in Monte Carlo methods, so before discussing Monte Carlo, we will review a few of the basic notions of probability theory. A distinction is usually made between discrete and continuous probability theory;

only continuous probability theory is discussed here, since it is most relevant for topics covered in this thesis.

3.2.1 Random Variables

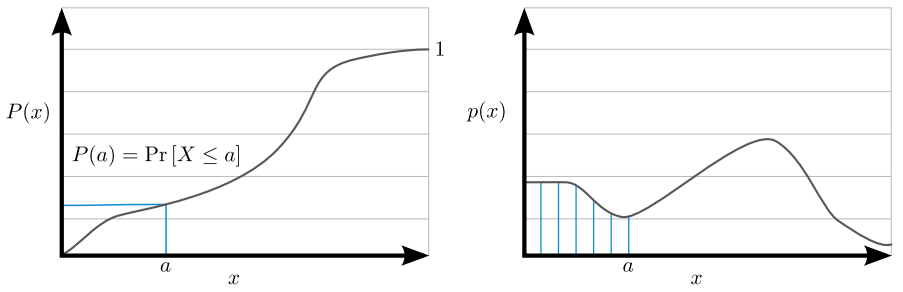
We begin with the definition of a random event: An event is a set of outcomes each assigned a probability. If each event is also assigned a real number, the result is called a *random variable*.

A cumulative distribution function (CDF) is a function that describes the distribution of probability of a random variable. It is written

$$P(x) = \Pr[X \leq x].$$

As illustrated in Figure 3.1, this is understood to mean the probability that the random variable X takes on a value less than or equal to x.

Figure 3.1: The cumulative distribution function (left) and the probability density function (right) for the 1D case.



A related concept is the probability density function (PDF), which describes the density of probability at each point in the sample space for a given random variable. It is closely related to the CDF, since

$$p(x) = \frac{\mathrm{d}P}{\mathrm{d}x}(x),$$

assuming the CDF is differentiable. Since the CDF is a non-decreasing function of x, the PDF must be a non-negative function of x. The PDF also has the normalization property

$$\int_{-\infty}^{\infty} p(x)\ \mathrm{d}x = P(\infty) - P(-\infty) = 1.$$

A special case is a *uniform* PDF, which is a PDF that is constant everywhere.

If $P(x)$ is continuous, X is called a *continuous random variable*. This implies that $\Pr[X = a] = 0$. On the other hand, if $P(x)$ is discontinuous, probability will concentrated at a set of discrete points. At these points the PDFs can be described using a *Dirac delta function*,

$$\delta(x) = \begin{cases} \infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

which, like the PDF, also has the normalizing property

$$\int_{-\infty}^{\infty} \delta(x)\, \mathrm{d}x = 1.$$

As a result PDFs are generally unbounded.

3.2.2 Expected Values and Moments

The expected value of a random variable $Y = f(X)$, or simply the expectation or mean, is given by

$$\begin{aligned} \mathrm{E}[Y] &= \int_{-\infty}^{\infty} f(x)\, \mathrm{d}P(x) \\ &= \int_{-\infty}^{\infty} f(x)\, p(x)\, \mathrm{d}x, \end{aligned} \tag{3.1}$$

assuming $P(x)$ is differentiable. The expected value is also called the first moment of X, and often denoted μ.

From the definition, it is easy to see that the expected value is linear

$$\mathrm{E}[aX + bY] = a\,\mathrm{E}[X] + b\,\mathrm{E}[Y],$$

for any constants a and b.

The moment of order n of X is given by

$$\mathrm{E}[X^n], \quad n = 1, 2, \ldots$$

Of particular interest are the central moments

$$\mathrm{E}[(X - \mu)^n] = \int_{-\infty}^{\infty} (x - \mu)^n\, p(x)\, \mathrm{d}x.$$

The second central moment is called the *variance*, and usually denoted

$$\sigma^2 = \mathrm{Var}\left[X\right] = \mathrm{E}\left[\,(X-\mu)^2\,\right]. \tag{3.2}$$

It has the property

$$\mathrm{Var}\left[aX+b\right] = a^2\,\mathrm{Var}\left[X\right],$$

for any constants a and b.

A convenient way of computing the variance can be found by expanding Equation 3.2

$$\begin{aligned}\mathrm{Var}\left[X\right] &= \mathrm{E}(X^2 - 2\mu X + \mu^2) = \mathrm{E}\left[X^2\right] - 2\mu\,\mathrm{E}\left[X\right] + \mu^2 \\ &= \mathrm{E}\left[X^2\right] - \mu^2 = \mathrm{E}\left[X^2\right] - \mathrm{E}\left[X\right]^2.\end{aligned}$$

The standard deviation, σ, which is the square root of the variance, is a useful measure of the dispersion of the random variable. It is also referred to as the root mean square error (RMSE).

A given moment is said to exist if $\mathrm{E}\left[X^n\right] < \infty$, that is, if it is finite. If $\mathrm{E}\left[X^n\right]$ is finite for a given n, then for $k \leq n$, $\mathrm{E}\left[X^k\right]$ is also finite. Conversely, if $\mathrm{E}\left[X^n\right]$ does not exist, then for $k \geq n$, $\mathrm{E}\left[X^k\right]$ does not exist either.

3.3 Monte Carlo Integration

Suppose we must solve the following definite integral,

$$F = \int_\Omega f(x)\ \mathrm{d}x,$$

for some possible multidimensional domain Ω. Assume the integrand $f(x)$ is a function of the form $\mathbb{R} \mapsto \mathbb{R}^n$ and F is the unknown parameter, or quantity of interest, called the *estimand*. In order to solve this integral using Monte Carlo integration, we first express F as an expectation,

$$F = \int_\Omega \frac{f(x)}{p(x)}\,p(x)\ \mathrm{d}x, \tag{3.3}$$

with respect to a probability density function p. Writing the problem this way makes it clear how to transform the original deterministic problem into a form suitable for Monte Carlo, since according to Equation 3.1, this is actually the expected value of a random variable. Or put another way: $\mathrm{E}\left[\frac{f(X)}{p(X)}\right] = F$.

This means we can approximate F using random sampling. To do this we generate N independent realizations of the random variable X, called sample points, or simply samples, $X_1, \ldots, X_N$ with density p, and compute the empirical average

$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}. \tag{3.4}$$

Since F_N "on average" provides a usable approximation of F, it is a valid *estimator* of F and F_N for a particular set of random variables $X_1, \ldots, X_N$ is called an *estimate* of F. To see why F_N is correct on average, note that F_N has the property that its expected value is equal to the correct result, since

$$\begin{aligned} \mathrm{E}\left[F_N\right] &= \mathrm{E}\left[\frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}\right] \\ &= \frac{1}{N} \sum_{i=1}^{N} \int_\Omega \frac{f(x)}{p(x)}\, p(x)\, \mathrm{d}x \\ &= \int_\Omega f(x)\, \mathrm{d}x = F. \end{aligned}$$

In general, an estimator is simply a function of the form

$$F_N = F_N(X_1, \ldots, X_N),$$

which can be used to estimate some unknown parameter, such as the mean. Consequently an infinite number of estimators exist for solving a given problem. In order to compare estimators, so that we can choose the best one, the properties of estimators is discussed below.

First, the *error* of an estimator is simply the difference between the estimate and the true value, $F_N - F$. The expected value of the error is called the *bias* of the estimator,

$$\beta\left[F_N\right] = \mathrm{E}\left[F_N - F\right].$$

An estimator is said to be unbiased if $\beta\left[F_N\right]$ is zero for all sample sizes, i.e.

$$\mathrm{E}\left[F_N\right] = F, \quad \text{for all } N \geq 1.$$

A less strict requirement than that of unbiasedness, is that an estimator should be *consistent*. A consistent estimator only converges to the mean in the limit

$$\Pr\left[\lim_{N\to\infty} F_N = F\right] = 1.$$

A sufficient condition for consistency is that both bias and variance go to zero in the limit

$$\lim_{N\to\infty} \beta\left[F_N\right] = \lim_{N\to\infty} \mathrm{Var}\left[F_N\right] = 0.$$

Unbiased estimators are generally preferred in Monte Carlo. The reason is that one often used measure of quality of a given estimator is its mean squared error,

$$\mathrm{MSE}(F_N) = \mathrm{E}[(F_N - F)^2]$$

For biased estimators the MSE can be written as the sum of the variance and bias

$$\mathrm{MSE}(F_N) = \mathrm{Var}\left[F_N\right] + \beta\left[F_N\right]^2.$$

Unfortunately this means that to estimate the MSE we need to know the bias, which we generally do not. For unbiased estimators we do not have this problem, since in this case

$$\mathrm{MSE}(F_N) = \mathrm{Var}\left[F_N\right].$$

This means that to get an error estimate, all we need to do is to estimate the variance.

As discussed above, low variance is a desirable property of an estimator. However, a property that is just as important is the speed of the estimator; that is how long it takes to generate a sample. This means that to evaluate a given estimator, both the variance and the temporal aspect must be taken into account, and this is exactly what is embodied in the notion of the *efficiency* of an estimator

$$\epsilon\left[F_N\right] = \frac{1}{\mathrm{Var}\left[F_N\right]\ \mathrm{T}\left[F_N\right]},$$

where T[·] is the time it taken to generate the samples.

Although unbiased estimators should be preferred, biased estimators should still be considered in some cases. Sometimes it is possible to construct biased estimators that are vastly more efficient than their unbiased counterparts. In such cases biased estimators should be preferred as long as they are consistent and it is possible to estimate or bound the bias. In other cases it is simply not possible to construct an unbiased estimator. This typically happens if the problem involves a ratio of integrals, in which case the estimator will always be biased.

3.4 Sampling Random Variables

To perform Monte Carlo computations, random numbers are required. True random numbers are rarely used, since producing these in large quantities is impractical. Instead, *pseudorandom* numbers, produced by a random number generator, are used. Though not truly random, since they are produced by an algorithm, and thus completely deterministic, these numbers are still suitable for Monte Carlo, since they have the same statistical properties as true random numbers.

Countless random number generators have been suggested in the Monte Carlo literature over the years. The Mersenne Twister, described in Saito and Matsumoto [2006], is a popular choice, and is also the random number generator used to produce the results in this thesis.

The Inversion Method

Often, random numbers with distributions different from uniform are desired. This happens for instance with some of the variance reduction methods discussed later in this chapter. The *inversion method*, described by Kalos and Whitlock [1986], is a general approach that makes it possible to generate random numbers distributed according to an arbitrary PDF. The only requirement is that the corresponding CDF must be continuous and invertible.

To make this more concrete, let $p(x)$ be a PDF with continuous CDF and let $P^{-1}(x)$ be its inverse, as illustrated in Figure 3.2. If ξ is a uniform random variable in $[0; 1]$, then $X = P^{-1}(\xi)$ is a random variable distributed according to $P(x)$.

To see why X is distributed according to $P(x)$, consider that

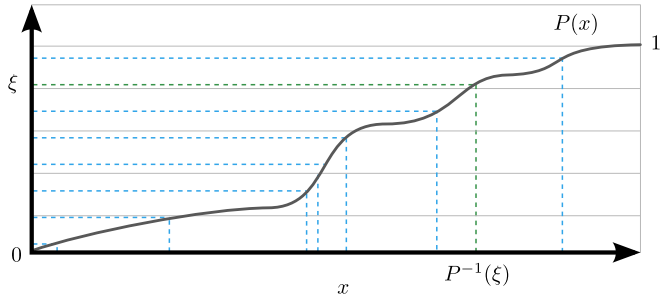
$$\Pr[\, X \leq x \,] = \Pr\left[\, P^{-1}(\xi) \leq x \,\right] = \Pr[\, \xi \leq P(x) \,] = P(x),$$

and recall that $P(x)$ is a non-decreasing function of x.

If inverting the CDF is not an option, a discrete approximation of the CDF can always be made. This is of course also an option if the PDF is discrete to begin with. Evaluating the inverse CDF can then be done using a binary search in $\mathrm{O}(\log_2 N)$ time for N sample points.

The inversion method can also be used in the multi-dimensional case. For instance, in 2D we might want to generate random variables (X, Y) distributed

Figure 3.2: The inversion method. Uniform random variables, ξ, are mapped to $X = P^{-1}(\xi)$ so that the distribution of X becomes $P(x)$ in the limit.



according to $p(x, y)$. To do so we first compute the marginal density

$$p_X(x) = \int p(x, y)\ \mathrm{d}y.$$

We can then sample X as $x = P_X^{-1}(\xi_1)$. Once we know X, the problem is reduced to 1D, and we can compute the conditional density for the corresponding slice,

$$p_Y(y|x) = \frac{p(x, y)}{p_X(x)}$$

and sample Y as $y = P_Y^{-1}(\xi_2|x)$. This procedure can be generalized to any number of dimensions.

3.5 Convergence Rates

As noted in the beginning, the convergence rate of Monte Carlo is $\mathrm{O}(N^{-\frac{1}{2}})$. To see why this is the case, consider first the variance of the estimator F_1. For convenience let $w(x) = \frac{f(x)}{p(x)}$, then

$$\mathrm{Var}\left[F_1\right] = \mathrm{Var}\left[w(X)\right].$$

Similarly, the variance of F_N is given by

$$\begin{aligned}\operatorname{Var}[F_N] &= \operatorname{Var}\left[\frac{1}{N}\sum_{i=1}^{N} w(X_i)\right] \\ &= \frac{1}{N^2}\operatorname{Var}\left[\sum_{i=1}^{N} w(X_i)\right] = \frac{1}{N^2}\sum^{N}\operatorname{Var}[w(X)] \\ &= \frac{1}{N}\operatorname{Var}[F_1],\end{aligned}$$

which means that the variance decreases linearly with $\frac{1}{N}$. The standard deviation, which can be seen as a measure of integration error, is then given by

$$\sigma[F_N] = \frac{1}{\sqrt{N}}\sigma[F_1],$$

which is the reason for the $\mathrm{O}(N^{-\frac{1}{2}})$ convergence rate of Monte Carlo.

The *Chebychev inequality* can be used to get a probabilistic bound on absolute error. It is so important that it is suggested by Kalos and Whitlock [1986] to name this inequality the first fundamental theorem of Monte Carlo. It is given by

$$\Pr\left[|F_N - F| \geq \sqrt{\frac{\operatorname{Var}[F_N]}{\delta}}\right] \leq \delta,$$

where δ is a small positive number. It tells us that the probability of generating an estimate that deviates greatly from the true mean can be made arbitrarily small simply by using enough samples.

The distribution of the mean of a random variable can be investigated using the *central limit theorem*. This theorem states that for a given number of samples, N, the distribution of F_N follow some PDF, and as $N \to \infty$, this distribution converges to the normal distribution. Following Veach [1997], this can be written

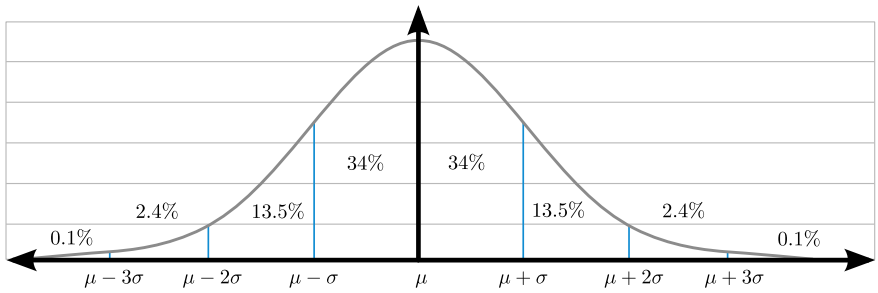
$$\lim_{N\to\infty}\Pr\left[F_N - F \leq t\frac{\sigma[F_1]}{\sqrt{N}}\right] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{t}\exp\left(\frac{-x^2}{2}\right)\mathrm{d}x.$$

Rearranging terms gives the following more convenient expression

$$\Pr\left[\,|F_N - F| \geq t\,\sigma[F_N]\,\right] = \sqrt{\frac{2}{\pi}}\int_{t}^{\infty}\exp\left(\frac{-x^2}{2}\right)\mathrm{d}x.$$

As shown in Figure 3.3, this means that given N is large enough for the Central Limit Theorem to apply, the so-called 68-95-99.7 rule applies. I.e., the probability that F_N is within one, two, or three standard deviations of F is 68%, 95%, and 99.7%, respectively.

Figure 3.3: The central limit theorem states that the distribution of the sum of N random variables each with an arbitrary PDF with finite mean, μ, and variance σ^2 approaches the normal distribution as $N \to \infty$. This means that the probability that the sum is within one, two, or three standard deviations of the true mean is 68%, 95%, and 99.7%, respectively.



This makes the central limit theorem a very powerful tool for investigating the distribution of F_N. However, one problem limiting the use of the theorem is that it only applies in the limit and determining exactly how large N must be for a given problem is generally nontrivial. A rule of thumb is presented by Kalos and Whitlock [1986] for when using the theorem is substantially satisfied, and it is given by

$$|\mu_3| \ll \sigma^3 \sqrt{N},$$

where μ_3 is the third central moment. If no such analysis is performed, and the central limit theorem is simply used without regards to sample size, the reported errors should be considered optimistic.

In the above discussion, we assumed that the variance existed. However, this is not always the case. If not, we need the *law of large numbers* (strong form), which states that if we sample a random variable with finite mean, like

$$F_N = \frac{1}{N} \sum_{i=1}^{N} w(X_i).$$

then F_N converges almost surely to the mean, F,

$$\Pr\left[\lim_{N\to\infty} F_N = F\right] = 1.$$

This is true even if the variance does not exist, although convergence in that case will be slower.

3.6 Blind Monte Carlo

Over the years numerous techniques have been developed for reducing variance in Monte Carlo computations. These techniques work by determining sampling locations more intelligently so that variance in the final estimate is reduced.

Recall that one of the most important goals when designing estimators is to maximize their efficiency

$$\epsilon[F_N] = \frac{1}{\mathrm{Var}[F_N]\ \mathrm{T}[F_N]}.$$

Since the use of variance reduction techniques typically requires more complicated computations, the time required to generate each sample is usually increased. That is, as $\mathrm{Var}[F_N]$ decreases, $\mathrm{T}[F_N]$ typically increases. The obvious implication of this is that it is only advantageous to use a variance reduction technique on a given problem if the relative decrease in variance is larger than the relative increase in computation time.

Unfortunately it can be hard to determine in advance whether using a variance reduction method on a given problem is an advantage. One solution is to perform trial runs with each estimator. If the first estimator produces a variance of σ_1^2 in t_1 seconds and the second σ_2^2 in t_2 seconds, an informed decision can be made based on their *relative efficiency*, which can be computed as the ratio $\frac{\sigma_1^2 t_1}{\sigma_2^2 t_2}$.

A distinction is made by Glassner [1994] between *blind* Monte Carlo and *informed* Monte Carlo variance reduction methods, that we will also use here. Blind Monte Carlo methods, which are covered in the remainder of this section, are methods which require no a priori knowledge about the integrand. In contrast, informed Monte Carlo methods use some knowledge of the integrand to construct low variance estimators. These methods are covered in the next section.

3.6.1 Crude Monte Carlo

For completeness, we will first consider the case of uniform sampling. Consider the integral $F = \int_{[0,1]^d} f(x)\ \mathrm{d}x$, where the domain is the d-dimensional hypercube. If this integral is estimated using independent, uniformly distributed random variables, then

$$F_N = \frac{1}{N} \sum f(X_i)$$

is an unbiased estimator of F and is called a *crude Monte Carlo* estimator. This is obviously just a special case of Equation 3.4 with $p(X_i) = 1$.

3.6.2 Rejection Sampling

Often in Monte Carlo we would like to generate samples distributed according to some function $f(x)$, which is not necessarily normalized. Unfortunately, if $f(x)$ is complicated or defined on some irregular domain, it may be difficult to do so. In such case the method of *rejection sampling*,[1] first proposed by von Neumann [1951], can be used.

The basic idea of rejection sampling is as follows. Imagine we wish to generate samples according to $f(x)$, but this function cannot be sampled from directly. Instead assume we have a simpler function, $q(x)$, that we can generate samples from, and a constant M, such that the envelope property is satisfied, i.e. $f(x) \leq M\, q(x)$ for all x.

Then, we first generate a tentative sample X' from $q(x)$. We then generate a uniform random number ξ in $[0; 1]$. If $\xi \leq \frac{f(X')}{M\, q(X')}$, X' is accepted as the next sample; otherwise the procedure is repeated until a sample that passes the test is found. Algorithm 3.1 shows the procedure in pseudo code.

Algorithm 3.1: Rejection sampling algorithm. Generate a sample distributed according to $f(x)$ using the envelope function $M\, q(x)$.

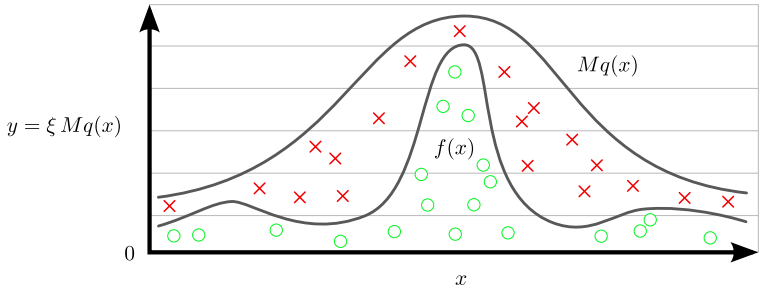
```
loop
    X' ~ q(x) {generate tentative sample X' from q(·)}
    ξ ~ 𝕌(0, 1) {sample uniform random number in [0, 1]}
    if ξ ≤ f(X') / (M q(X')) then
        X ← X' {accept tentative sample}
        break
    end if
end loop
```

An alternative way of thinking about rejection sampling is to consider the sampling procedure as increasing the dimension by one. This idea is illustrated in Figure 3.4 for the 1D case. In this case, to find the 2D coordinate (x, y) of the tentative sample, we first sample x from $q(x)$. We then sample a random number ξ and compute the y coordinate as $y = \xi\, M\, q(x)$. If $y < f(x)$ the sample is accepted; otherwise a new tentative sample is generated.

[1] Sometimes also referred to as *hit-or-miss* sampling.

The biggest problem with rejection sampling is that if $M\,q(x)$ is chosen unwisely many tentative samples might have to be generated to get a single accepted sample. Looking at Figure 3.4 it is clear that to reduce the number of rejected samples $M\,q(x)$ should follow $f(x)$ as closely as possible. I.e. the area between $f(x)$ and $M\,q(x)$ should be kept as small as possible.

Figure 3.4: Illustration of the rejection sampling algorithm. Green circles represent accepted samples and red crosses rejected samples.



The expected number of tentative samples required to generate a single accepted sample can be found as the ratio

$$n = \frac{M \int q(x)\ \mathrm{d}x}{\int f(x)\ \mathrm{d}x}.$$

If we assume $f(x)$ is normalized ($q(x)$ is always normalized) then this is reduced to $n = M$. This means that if M is large and evaluating $f(x)$ is expensive, rejection sampling can be very inefficient. In such cases the Metropolis-Hastings algorithm, presented in Section 3.8.2, should be considered instead, since it is able to also utilize information from the rejected samples.

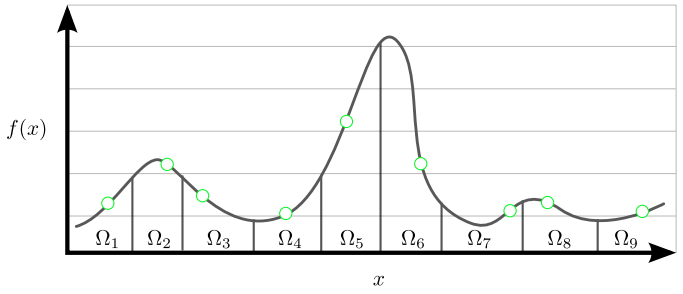
3.6.3 Stratified Sampling

Techniques, such as crude Monte Carlo, have the problem that the resulting samples only follow the desired distribution, e.g. uniform, in the limit. This means that when using relatively few samples generated independently chances are that these samples will often tend to clump together rather than cover the domain uniformly. This is unfortunate, since this will manifest itself as increased variance.

One way to reduce this clumping is to use the technique of *stratified sampling*. As shown in Figure 3.5, the intuition behind stratified sampling is that if we

split the domain into k disjoint regions, and place a few samples in each region, then by construction the samples are forced to cover the entire domain, and clumping should be reduced.

Figure 3.5: Illustration of the stratified sampling algorithm. The domain of f, Ω, is divided up into k disjoint regions, $\Omega_1, \Omega_2, \ldots, \Omega_k$, and a random sample is taken in each region.



Each of these disjoint regions is called a *stratum* and their union is the entire domain

$$\bigcup_{i=1}^{k} \Omega_i = \Omega.$$

Following Veach [1997], if we assume the domain is the d-dimensional unit cube, $\Omega = [0;1]^d$, and that we place n_i samples in each stratum with uniform probability, then the estimator is of the form

$$F' = \sum_{i=1}^{k} v_i F_i,$$

where v_i is the volume of Ω_i and the estimator for each stratum is

$$F_i = \frac{1}{n_i} \sum_{j=1}^{n_i} f(X_{i,j}),$$

where $X_{i,j}$ is the jth independent sample in the ith stratum. The variance of this estimator is

$$\mathrm{Var}\,[F'] = \sum_{i=1}^{k} \frac{v_i^2}{n_i} \sigma_i^2,$$

where σ_i^2 is the variance of F_i.

If we assume that the number of samples in each region is proportional to volume, $n_i = v_i N$, where N is the total number of samples, $N = \sum_{i=1}^{k} n_i$, then the variance can be written as

$$\text{Var}\left[F'\right] = \frac{1}{N} \sum_{i=1}^{k} v_i \sigma_i^2. \tag{3.5}$$

For comparison, as shown by Veach [1997, pg. 51], the variance of the normal (non-stratified) estimator is

$$\text{Var}\left[F\right] = \frac{1}{N} \left[\sum_{i=1}^{k} v_i \sigma_i^2 + \sum_{i=1}^{k} v_i (\mu_i - I)^2 \right],$$

where μ_i is the average value in Ω_i and I is the average in Ω. Compared to Equation 3.5, it can be seen that stratified samples can never increase variance. However, variance is only reduced if strata can be chosen with means that differ from the function average I. Also, even though variance is never increased, efficiency can still be reduced with stratified sampling if the time to generate each sample is increased, though the simple nature of stratified sampling makes this unlikely to be a problem in practice.

The convergence rate of stratified sampling for a certain class of functions is discussed by Mitchell [1996]. He shows that when using stratified sampling on functions that obey a Lipschitz smoothness condition the variance is $\text{O}(N^{-1-2/d})$, where N is the number of samples and d is the number of dimensions. Compared to the normal convergence rate of $\text{O}(N^{-1})$, it is clear that stratified sampling can be a great advantage for low dimensional problems (e.g. $\text{O}(N^{-3})$ convergence in 1D), but that this benefit is quickly lost as the dimension increases.

For high dimensional problems, using stratified sampling is unpractical, since the number of samples required grows too fast. For instance, if 100 samples are used per dimension, the growth is $\text{O}(100^d)$, something often referred to as the curse of dimensionality. However, stratified sampling can still be used for many-dimensional problems in some cases. The idea is first to identify the dimensions that are responsible for the majority of variation in the integrand and then only stratify these, rather than all the dimensions. Kalos and Whitlock [1986] report that using this strategy can significantly reduce variance.

3.7 Informed Monte Carlo

In the last section we covered blind Monte Carlo methods: variance reduction methods that did not use any information about the integrand. In this section informed Monte Carlo methods are presented. These methods use information about the integrand (full or partial) to guide the placement of samples to important regions in the domain of the integrand.

3.7.1 Importance Sampling

Consider again Equation 3.3,

$$F = \int_\Omega \frac{f(x)}{p(x)}\, p(x)\ \mathrm{d}x.$$

As discussed in the previous sections, in Monte Carlo, we estimate such integrals using estimators that are often of the form

$$F_N = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}.$$

The variance of F_N depends on both $f(x)$ and $p(x)$. To reduce the variance, note that we can not change $f(x)$, but are free to change $p(x)$ within the previously mentioned restrictions. This begs the question of what choice of $p(x)$ produces an estimator with minimum variance for a given $f(x)$.

This question is answered by Kalos and Whitlock [1986, pg. 92]. First consider that the variance of F is

$$\mathrm{Var}\,[F] = \int_\Omega \left(\frac{f(x)}{p(x)}\right)^2 p(x)\ \mathrm{d}x - F^2.$$

Ideally we would like $\mathrm{Var}\,[F]$ to be as small as possible by choosing a suitable $p(x)$. This can be achieved by minimizing $\mathrm{Var}\,[F]$ with respect to $p(x)$ using the method of Lagrange multipliers. To do this we need to find a scalar λ such that

$$L(p) = \int_\Omega \frac{f^2(x)}{p(x)}\ \mathrm{d}x + \lambda \int_\Omega p(x)\ \mathrm{d}x$$

is minimized. To find the minimum we first differentiate with respect to p and set the result to zero,

$$\begin{aligned} 0 &= \frac{\partial}{\partial p}\left[\int_\Omega \frac{f^2(x)}{p(x)}\ \mathrm{d}x + \lambda \int_\Omega p(x)\ \mathrm{d}x\right] \\ &= -\frac{f^2(x)}{p^2(x)} + \lambda. \end{aligned}$$

It is now straight forward to solve for $p(x)$

$$p(x) = \frac{1}{\sqrt{\lambda}}|f(x)|,$$

which means that the ideal $p(x)$ must be proportional to $f(x)$. To find the constant of proportionality recall that $p(x)$ is a PDF, so it must be normalized. This means that

$$\begin{aligned} 1 &= \int_\Omega \frac{1}{\sqrt{\lambda}} f(x)\ \mathrm{d}x \\ &= \frac{F}{\sqrt{\lambda}}, \end{aligned}$$

and the optimal PDF is then given by

$$p(x) = \frac{f(x)}{F} = \frac{f(x)}{\int_\Omega f(x)\ \mathrm{d}x}.$$

To check that this is indeed the optimal choice of $p(x)$, we can compute the variance

$$\begin{aligned} \mathrm{Var}\left[F\right] &= \int_\Omega \left(\frac{f(x)}{f(x)/F}\right)^2 f(x)/F\ \mathrm{d}x - F^2 \\ &= F^2 - F^2 = 0, \end{aligned}$$

which means that no choice of $p(x)$ can be better.

Unfortunately, direct application of this method is impossible, since we need to know F, which is the quantity we are trying to find in the first place. In addition, we can not hope to draw samples from $f(x)$, since its complicated nature is what made us resort to Monte Carlo methods. However, we can still use the general principle, if we note that the reason this "perfect" sampling resulted in zero variance is because the ratio $f(x)/p(x)$ is constant for any x. Since it is not possible to make this ratio absolutely constant, we must contend ourselves by picking a $p(x)$ that is only roughly proportional to $f(x)$ and that results in a $f(x)/p(x)$ that is only approximately constant.

This principle is called *importance sampling* and was first recognized by Marshall [1956]. It is arguably the most important variance reduction method. The name comes from the fact that when using importance sampling more samples are placed where the integrand is large, i.e. in the important regions of Ω.

In practice finding a suitable $p(x)$ is always a compromise, since on the one hand $p(x)$ should be simple enough so that it can be sampled from, e.g. using the inversion method. But on the other hand $p(x)$ should be as similar as possible to the complicated function $f(x)$, since this will result in a low variance estimator.

Multiple Importance Sampling

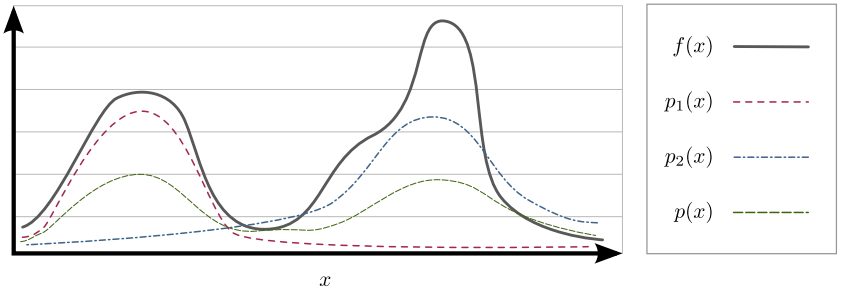
As discussed above, constructing a single PDF that matches the integrand well while still allowing for sampling is a hard problem. As it turns out, it is often the case that several candidate probability density functions are available. Individually these functions would typically only be a good match for the integrand in some subset of the domain, but taken together they could potentially be a good match in the entire domain.

This might happen if the integrand is a product of two or more simpler functions, such as if

$$F = \int_\Omega g(x)\, h(x)\, \mathrm{d}x,$$

in which case generating samples distributed according to either $g(x)$ or $h(x)$ might be possible. These PDFs could also be found in other ways. For instance, the bimodal function shown in Figure 3.6 could be sampled using a combination of samples from two normal distributions centered at each mode.

Figure 3.6: Generating samples distributed according to a complicated $f(x)$ is often not possible. Instead samples can be generated using the simpler functions $p_1(x)$ and $p_2(x)$ and combined using the framework of multiple importance sampling. The resulting *effective* PDF is $p(x)$, assuming the balance heuristic is used and that the same number of samples is taken from each technique.



To make this more general, consider again the integral

$$F = \int_\Omega f(x)\, \mathrm{d}x.$$

Imagine we have n sampling techniques, $p_1, p_2, \ldots, p_n$ and that we take $n_i > 0$ samples from each technique p_i. This leads to the samples $[X_{i,1}, X_{i,2}, \ldots, X_{i,n_i}]$

from each technique, and to a total of $N = \sum_{i=1}^{n} n_i$ samples. Then the following

$$F_N = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{f(X_{i,j})}{p_i(X_{i,j})} \tag{3.6}$$

is a simple unbiased estimator of F. The downside to this estimator is that its variance is potentially very high. To see why, consider again Figure 3.6. In this case it can happen that a sample generated from $p_1(x)$ falls in the right lobe of $f(x)$. This will only happen with low probability, so $p(x)$ will be small, but $f(x)$ will be large, which according to the principles of importance sampling, will lead to high variance. If we let $a_i = \frac{1}{(nn_i)^2}$, then the variance of this estimator can be written as

$$\operatorname{Var}[F_N] = a_1 \operatorname{Var}\left[\sum_{j=1}^{n_1} \frac{f(X_{1,j})}{p_1(X_{1,j})}\right] + \cdots + a_n \operatorname{Var}\left[\sum_{j=1}^{n_n} \frac{f(X_{n,j})}{p_n(X_{n,j})}\right].$$

I.e. the variance of the estimator is simply a weighted sum of the variances of each of the individual techniques. This means that if one or more techniques results in high variance, then $\operatorname{Var}[F_N]$ will also be high.

Ideally we would like to combine samples from different sampling techniques in a way that preserves their relative strengths, while reducing their influence where the techniques perform poorly. An effective way of doing this is to use the technique of *multiple importance sampling* described by Veach and Guibas [1995] (with more details given in Veach [1997, chp. 9]).

The basic idea of multiple importance sampling is that rather than using a constant weight for samples from a given technique, better results can be achieved if the weight is allowed to depend on the sample location and the other sampling techniques. This idea is embodied in the *multi-sample estimator*

$$F_N = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})},$$

which is an unbiased estimator of F as long as

1. $\sum_{i=1}^{n} w_i(x) = 1$ whenever $f(x) \neq 0$ and
2. $w_i(x) = 0$ whenever $p_i(x) = 0$.

Several weighting functions are suggested by Veach and Guibas [1995]. A weighting function known to work well is the *balance heuristic*. It is given by

$$w_i(x) = \frac{n_i\, p_i(x)}{\sum_k n_k\, p_k(x)}.$$

Note that $w_i(x)$ depends on x and on $p_k(x)$ for $k \neq i$. This means that to compute $w_i(x)$ it must be possible to evaluate the PDFs for all the other techniques. To see the intuition behind this weighting function, it can be inserted into the multi-sample estimator

$$\begin{aligned} F_N &= \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} \left(\frac{n_i\, p_i(x)}{\sum_k n_k\, p_k(x)} \right) \frac{f(X_{i,j})}{p_i(X_{i,j})} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n_i} \frac{f(X_{i,j})}{\sum_k n_k\, p_k(x)} \\ &= \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \frac{f(X_{i,j})}{\sum_k c_k\, p_k(x)}, \end{aligned}$$

where $c_k = n_k/N$ is the fraction of samples from the kth technique. Essentially, this means that compared to the regular estimator (Equation 3.6), $p_i(x)$ is replaced with its expected value, $p(x) = \sum_k c_k\, p_k(x)$, which is the actual mixture density used to generate samples.

To see why using multiple importance sampling reduces variance, we again return to Figure 3.6. In this example we saw that if a sample drawn from $p_1(x)$ ended up in the right mode of $f(x)$, this would result in high variance when using the original estimator, since $f(x)/p_i(x)$ was far from constant. When using the balance heuristic, $p_i(x)$ is replaced with $p(x) = \sum_k c_k\, p_k(x)$, which is a much better match for $f(x)$. As result, the ratio $f(x)/p(x)$ is much closer to constant, and thus in general has lower variance. A proof that the balance heuristic always performs within an additive constant of the optimal combination strategy is given by Veach [1997, pg. 264].

A generalization of the balance heuristic is the *power heuristic*

$$w_i(x) = \frac{(n_i\, p_i(x))^\beta}{\sum_k (n_k\, p_k(x))^\beta}.$$

The exponentiation has the effect of sharpening the function, which can be advantageous for some low variance problems. A value of $\beta = 2$ has been suggested to work well on many problems.

The *cutoff heuristic* can also sometimes outperform the balance heuristic on low variance problems. It works by ignoring contributions that occur with low probability and is given by

$$w_i(x) = \begin{cases} 0, & \text{if } n_i\, p_i(x) < \kappa \\ \frac{n_i\, p_i(x)}{\sum_k \{n_k\, p_k(x) \mid n_k\, p_k(x) > \kappa\}} & \text{otherwise} \end{cases}$$

where κ is the maximum of $\alpha\, n_k\, p_k(x)$ and α is a parameter that controls where the cutoff occurs. The *maximum heuristic* only considers samples from the most likely technique, and ignores all other samples.

3.8 Markov Chain Monte Carlo

In the previous sections we saw how we could estimate an integral using Monte Carlo integration. The basic idea was that an estimator of integrals of the form

$$F = \int_\Omega f(x)\ \mathrm{d}x,$$

could be formed as the expected value of a random variable F_N,

$$\mathrm{E}\left[F_N\right] = \mathrm{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}\right],$$

with the random variables $X_1, X_2, \ldots, X_N$ distributed according to the PDF, $p(x)$. Then, since the samples are assumed to be independent, the law of large numbers told us that F_N would converge to F as $N \to \infty$ almost surely.

We also saw how the efficiency of the estimator depended crucially on the choice of $p(x)$, and why the ideal choice of PDF was $p(x) \propto f(x)$.

Unfortunately, generating independent samples distributed according to some possibly very complex PDF can be very challenging. However, if the independence requirement is relaxed, then any process that generates samples distributed according to $f(x)$ can be used. In particular, if the samples are generated from a Markov chain with stationary distribution $\pi^*(x) \sim f(x)$, then the resulting algorithm is said to be a Markov Chain Monte Carlo algorithm. As it turns out, this leads to a class of algorithms that can sample effectively from even very complicated PDFs.

The price of using dependent samples is that the methods for testing convergence presented in Section 3.5 can not be directly applied anymore, since they require independently generated samples. However, it is possible to work around this problem by running multiple experiments in parallel. Since the outcomes of these experiments will be independent, the central limit theorem can still be used to asses convergence.

The following is based on Gilks, Richardson, and Spiegelhalter [1995] and Robert and Casella [2005].

3.8.1 Markov Chains

A Markov chain is a special kind of stochastic process. They are named after the Russian mathematician Andrey Andreyevich Markov. What distinguishes them from other stochastic processes is that they have the *Markov property*, which states that if $\{X_1, X_2, X_3, \ldots\}$ is a sequence of random variables with this property, then

$$\Pr(X_{t+1} = x | X_t = x_t, \ldots, X_1 = x_1) = \Pr(X_{t+1} = x | X_t = x_t).$$

That is, the probability of choosing the next state X_{t+1} depends only on the current state X_t and thus not on the earlier history of the chain.

Markov chains are either continuous-time or discrete-time stochastic processes, $(X_t)_{t \in \mathbb{N}}$. Since continuous-time Markov chains are not used in Markov Chain Monte Carlo, this type is not considered further, and all chains are assumed to be discrete-time chains.

In the context of Markov Chains, the conditional probability density, $\Pr(X_{t+1} = x | X_t = x_t)$, which describes how the chain evolves, is called the *transition kernel* and gives the probability of moving from one state to another in one step.

All the possible values of x form the *state space* of the Markov chain. If the state space is finite, the chain is a discrete-state Markov chain and the transition kernel can be described with a transition matrix. In the continuous case, such as when $x \in \mathbb{R}^n$, the transition kernel must be a normalized conditional density,

$$\int \Pr(X_{t+1} = x | X_t = x_t) \, \mathrm{d}x = 1.$$

A chain is said to be time-homogeneous if

$$\Pr(X_{t+1} = x | X_t = x_t) = \Pr(X_t = x | X_{t-1} = x_t),$$

which basically means that the transition kernel does not change over time. Time-homogeneous chains are by far the most common, and also what is considered in this thesis.

In order for the distribution of (X_t) for a Markov chain to converge to a given *stationary distribution*, $\pi^*(x)$, the chain must satisfy three properties. These properties are that the chain must be *irreducible*, *aperiodic*, and *positive recurrent*.

Informally, a chain is said to be irreducible if for any starting point the chain can reach any other state with positive probability. This is essentially a form of

connectedness condition. More formally, a state x is said to be accessible from another state x_0 in t steps if

$$\Pr(X_t = x | X_0 = x_0) > 0,$$

for $t > 0$. If x_0 is also accessible from x, these two states are said to communicate with each other. A set of states that communicate with each other form a communicating class, and if the entire state space happens to be a single communicating class, then the chain is said to be irreducible.

The requirement of aperiodicity is necessary to ensure that the chain does not oscillate between two or more states in regular intervals. A state x is said to have period

$$k = \gcd \{t > 0 : \Pr(X_t = x | X_0 = x) > 0\},$$

if revisits to state x must occur in multiples of k time steps and gcd is the greatest common divisor. An irreducible chain is called aperiodic if $k = 1$ for any state.

Irreducibility ensures that a Markov chain will visit every region of the state space. However, this is a fairly weak property, as it does not specify how often any region will be visited. It turns that stronger statements can be made if chains are classified according to whether they are *transient* or *recurrent*.

A state is said to be transient if there is a non-zero probability that the chain will never revisit a given state. Let τ_x be the time of the first return to state x

$$\tau_x = \min \{t > 0 : X_t = x | X_0 = x\}.$$

Then a state is said to be transient if

$$\Pr(\tau_x = \infty) > 0.$$

If the state is not transient, it is recurrent. If all states are recurrent (transient), the chain is said to be recurrent (transient). An irreducible recurrent chain is called *positive recurrent* if the expectation of the revisit time to given state is finite

$$\mathrm{E}[\tau_x] < \infty,$$

for any x. Otherwise the chain is *null-recurrent*.

Together, a chain that is both aperiodic and positive recurrent is called *ergodic*. As stated above, an irreducible chain has a stationary distribution if and only if it is positive recurrent. In addition, the stationary distribution is unique and satisfies

$$\pi^*(x) = \int K(y \to x)\, \pi^*(y)\, \mathrm{d}y,$$

where $K(y \to x) = \Pr(X_{t+1} = x | X_t = y)$ is the transition kernel.

It is important to remember that this is only true in the limit. For instance, consider a chain with starting value $X_0 = x_0$. For this chain the distribution of X_0 is $\pi_0(x) = \delta(x - x_0)$. The continued evolution of the chain is described by the Chapman-Kolmogorov equation

$$\pi_{t+1}(x) = \int K(y \to x)\, \pi_t(y)\, \mathrm{d}y, \tag{3.7}$$

and it is only in the limit, as $t \to \infty$, that we have

$$\lim_{t \to \infty} \pi_t = \pi^*.$$

This means that the stationary distribution is the limiting distribution of π_t. This is true regardless of the choice of starting point x_0, as the chain forgets its starting value over time.

3.8.2 Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is a well known algorithm from computational physics. It was first described in a short paper by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller [1953], and is therefore also known as the $\mathrm{M(RT)}^2$ algorithm. It was later generalized by Hastings [1970].

As described above, the basic idea of Markov chain theory is to construct a chain using a given transition kernel. If the transition kernel satisfies certain conditions, the chain is known to have a unique stationary distribution. The Metropolis-Hastings algorithm, and Markov Chain Monte Carlo in general, turns this around. Instead a stationary distribution is given, and the goal is to construct a transition kernel that causes the Markov chain to converge to this distribution.

Finding such a transition kernel may seem impossible. However, a simple recipe exists for finding such functions based on the concept of *detailed balance*. Informally, detailed balance is the equilibrium condition given by

$$\pi^*(x)\, K(x \to y) = \pi^*(y)\, K(y \to x).$$

Since we are trying to sample from f, we want $\pi^*(x) = f(x)/F$, where F is the unknown normalizing constant $F = \int_\Omega f(x)\, \mathrm{d}x$. Substituting $f(x)$ for $\pi^*(x)$, we have

$$f(x)\, K(x \to y) = f(y)\, K(y \to x).$$

Intuitively this says that the system will favor moves from places where f is small to places where f is large, which makes sense, since we want $\pi^*(x) \propto f(x)$.

Following Chib and Greenberg [1995], to find a usable transition kernel, consider first what happens if a *tentative transition kernel*[2] $T(x \to y)$ is used, which does not satisfy the detailed balance condition. If this happens, we might have that

$$f(x)\, T(x \to y) > f(y)\, T(y \to x). \tag{3.8}$$

This means that we move too often from x to y and too rarely from y to x. To correct this, we introduce a function $\alpha(x \to y) \leq 1$, which gives the probability of actually performing the move. The transition kernel can then be written as

$$K(x \to y) = T(x \to y)\, \alpha(x \to y).$$

What this means is that we first generate a tentative sample, y, using $T(x \to y)$, which we then probabilistically accept or reject according to $\alpha(x \to y)$. As described in Algorithm 3.2, if the sample is accepted it becomes the next state of the chain; otherwise the old sample becomes the next state. This essentially makes the Metropolis-Hastings algorithm a form of rejection sampling, though with the important difference that unlike regular rejection sampling, as described in Section 3.6.2, the Metropolis-Hastings algorithm uses the framework provided by Markov chain theory

Algorithm 3.2: The Metropolis-Hastings algorithm. The initial sample could be chosen with uniform probability in Ω. The only requirement is that $f(x^{(0)}) > 0$.

$x^{(0)} \leftarrow$ {generate initial sample, ensure $f(x^{(0)}) > 0$}
for $i = 1$ to N **do**
  $y \sim T(x^{(i-1)} \to y)$ {generate sample from tentative transition kernel}
  $\xi \sim \mathbb{U}(0, 1)$ {sample uniform random number in $[0, 1]$}
  if $\xi < \alpha(x^{(i-1)} \to y)$ **then**
    $x^{(i)} \leftarrow y$ {accept tentative sample with probability $\alpha(x^{(i-1)} \to y)$}
  else
    $x^{(i)} \leftarrow x^{(i-1)}$ {reject proposed sample and stay at $x^{(i-1)}$}
  end if
end for

To determine $\alpha(x \to y)$ consider again Equation 3.8. In that example there were too few moves from y to x (or too many moves from x to y). One way to correct this is to set $\alpha(y \to x) = 1$, and $\alpha(x \to y) < 1$. As discussed later, this turns out to be a good strategy for making the chain converge quickly, since it maximizes

[2] In some texts called the proposal distribution.

the probability of accepting a move. Inserting this into the detailed balance equation, we get

$$\begin{aligned} f(x)\,T(x{\to}y)\,\alpha(x{\to}y) &= f(y)\,T(y{\to}x)\,\alpha(y{\to}x) \\ &= f(y)\,T(y{\to}x). \end{aligned}$$

Given this choice, the acceptance probability, $\alpha(x{\to}y)$, becomes

$$\alpha(x{\to}y) = \min\left[\frac{f(y)\,T(y{\to}x)}{f(x)\,T(x{\to}y)}, 1\right].$$

Note that the normalizing constant F in $\pi^*(x) = f(x)/F$ does not appear, as it has canceled out. This is fortunate, since estimating F might easily be as hard as solving the original problem.

If the tentative transition kernel is symmetric, $T(x{\to}y) = T(y{\to}x)$, the acceptance probability simplifies to

$$\alpha(x{\to}y) = \min\left[\frac{f(y)}{f(x)}, 1\right].$$

This is in fact the original algorithm proposed by Metropolis et al. in 1953. Using a symmetric transition kernel makes it a bit easier to see how the algorithm works: If $f(x) < f(y)$ the move is deterministically accepted and if $f(x) > f(y)$ the move is accepted with probability $f(y)/f(x)$. The important generalization of the algorithm to non-symmetric transition kernels was made by Hastings in 1970.

To verify that that the Metropolis-Hastings kernel does not disturb the stationary distribution, we can insert it into the Chapman-Kolmogorov equation (Equation 3.7)

$$\begin{aligned} \pi_{t+1}(x) &= \pi_t(x)\left[1 - \int_\Omega T(x{\to}y)\,\alpha(x{\to}y)\;\mathrm{d}y\right] + \int_\Omega \pi_t(y)\,T(y{\to}x)\,\alpha(y{\to}x)\;\mathrm{d}y \\ &= \pi_t(x) + \int_\Omega \Big[\pi_t(y)\,T(y{\to}x)\,\alpha(y{\to}x) - \pi_t(x)\,T(x{\to}y)\,\alpha(x{\to}y)\Big]\;\mathrm{d}y \\ &= \pi_t(x). \end{aligned}$$

The two terms on the right hand side of the first line of the equation is the probability of already being at x and not leaving and the second is the probability of transitioning to x from elsewhere. In the second line the integral vanishes, due to the detailed balance condition.

Startup Bias

Since the Metropolis-Hastings algorithm is only guaranteed to sample from $f(x)$ asymptotically, the first samples, $\{x_0, x_1, \ldots\}$ are not necessarily distributed according to π^*. If these samples are used anyway, the results will suffer from systematic error, something usually referred to as start up bias.

To avoid start up bias, it is necessary to allow the sampler to "burn in" before using any of the samples. Ideally this would mean that all samples generated before the chain reaches its stationary distribution are discarded. Unfortunately determining the burn-in period for given chain can be very difficult.

Random Walk and Independence Kernels

Two classical tentative transition kernels used in Markov Chain Monte Carlo are the random walk kernel and the independence kernel. When using the random walk kernel

$$T_{\text{r.w.}}(x \to y) = T(|x - y|),$$

the next state is sampled as a slightly perturbed, or mutated, version of the current state. I.e., $x^{(t+1)} = x^{(t)} + \epsilon$, where ϵ is typically a random variable with a (multivariate) normal distribution and constant covariance matrix. Note that these transition kernels are symmetric.

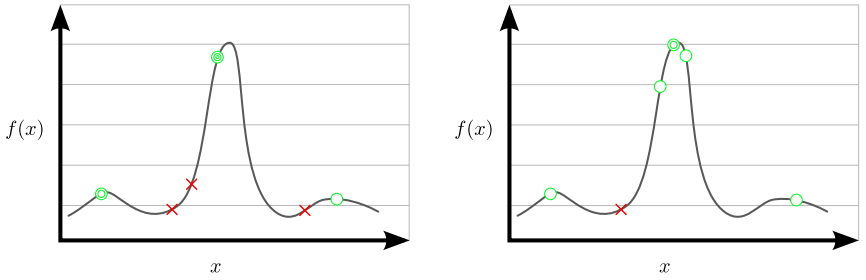
Independence transition kernels are of the form

$$T_{\text{ind.}}(x \to y) = T(y),$$

which, as the name implies, means that y is sampled independently of x. Independence kernels are often used in combination with random walk kernels to ensure a faster exploration of the state space.

Random walk transition kernels allow for local exploration of $f(x)$, which can be a great advantage if the integrand is complicated. For instance, consider an integrand which has its mass concentrated in some small subset of its high-dimensional domain. Since in general we have no a-priori knowledge of where these regions are, we can only hope to find them using random sampling. This means that when using independent samples, these high value regions are only sampled with low probability, which leads to a high variance estimator. As illustrated in Figure 3.7, random walk algorithms have the same low probability of finding samples in these important regions. However, once found, they will explore nearby samples, which will typically also tend to be important. This

Figure 3.7: Comparison of two different mutation strategies used for generating 6 samples from a simple 1D target density. Green circles signify accepted samples (concentric green circles imply repeated samples), and red crosses mean rejected samples. Independent samples generated using a uniform PDF (left) leads to many rejected samples and consequently highly correlated samples. A random walk (right) often performs better since it can perform local exploration of the function.



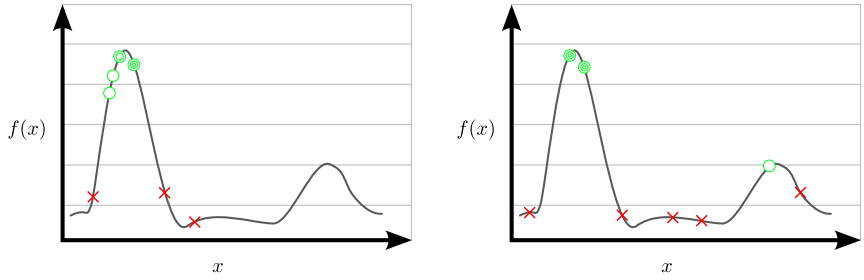
(a) Independence kernel. **(b)** Random walk kernel.

property of being able to preserve the sampling context means that the high cost of finding the initial sample can be amortized over multiple samples.

For a random walk algorithm to be effective, the transition kernel must be tuned to $f(x)$. In particular, for a random walk algorithm based on proposals with a normal distribution, the covariance matrix should be chosen so that the size of the mutations matches the size of features of $f(x)$. To see why this is important, consider what happens if the variance is chosen too small or too large, as shown in Figure 3.8a. If the variance is chosen too small, the acceptance rate will be high, but the samples will be highly correlated, since the mutations are small. On the other hand, if the variance is large, as is the case in Figure 3.8b, the mutations will also be large, and the proposed samples will tend to be in regions of low importance. This leads to a low acceptance rate, and consequently highly correlated samples, since the chain will not move at all.

Unfortunately target densities cannot be assumed to be unimodal, so finding a globally optimal mutation size is generally not possible. For instance, consider the 1D bimodal target density show in Figure 3.9. Here the modes are sufficiently dissimilar, so that a good mutation size for one of the modes will likely result in poor performance for the other. In Figure 3.9a the mutation size has been chosen to match the width of the left mode. Assuming the chain started in this mode, samples might very well only been generated in this mode, since the probability of "jumping" to the other mode is very small.

Figure 3.8: The effect of mutation size on mixing and acceptance rate. Very small mutations (left) give a high acceptance rate, but correlated samples and a poorly mixing chain. Conversely, very large mutations (right) give a low acceptance rate, and thus also highly correlated samples, due to the many rejections. As a result, the optimal mutation size usually produces a medium acceptance rate.



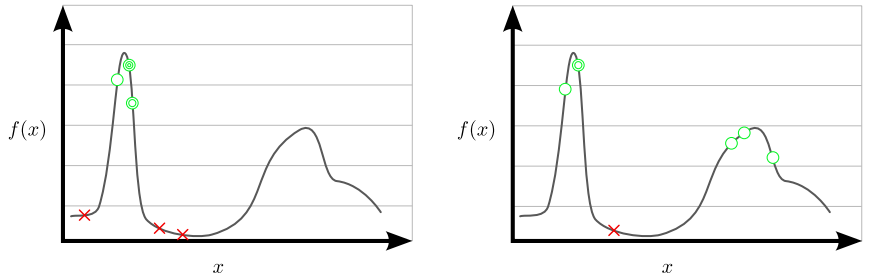
(a) Small mutations. **(b)** Large mutations.

Such chains are called *slowly mixing*. Slowly mixing chains still converge to their stationary distribution in the limit (assuming they are ergodic). However, the number of samples (and thus the time required) will be very high. As a consequence, chains which quickly explore all the state space, rather than getting stuck in some region are clearly preferable. Such chains, usually called *rapidly mixing* chains, not only have shorter burn-in period, but also generate less correlated samples, which reduces the error in the estimate.

To overcome the problem that random walk kernels can get stuck in local modes, it is customary to combine these kernels with the independence kernel. An example of this is given in Figure 3.9b. This means that before each new sample is generated, a random choice is made between which of the two kernels to use. If the random walk kernel is chosen, the sample space will be explored near the current sample, which has the advantage of preserving the sampling context. On the other hand, if the independence kernel is chosen, the chain will explore new regions of the sample space and the ergodicity of the chain will be ensured.

This still leaves the problem of finding a suitable mutation size for an arbitrary target density. Much research has been devoted to developing techniques for solving this problem. For instance, Gelman, Roberts, and Gilks [1995] found that for a certain class of target distributions, the asymptotically optimal acceptance rate is approximately 0.234 as the dimension $d \to \infty$. This means that to find the best mutation size, the acceptance rate of a pilot run can be monitored. If the acceptance rate is higher than 0.234 then this means that the proposed mutations are too small and that the variance of the proposal function should

Figure 3.9: Comparison of using the random walk kernel alone with using the random walk kernel together with the independence kernel. The random walk kernel tends to get stuck in local modes. Supplementing the local moves with independent samples has the desirable effect of improving the mixing of the chain and ensuring ergodicity.



(a) Random walk kernel. **(b)** Both kernels.

be increased. Conversely, if the acceptance rate is too low, the variance should be lowered.

3.9 Summary

In this chapter we have provided a general introduction to Monte Carlo methods. Variance reduction methods, such as stratified sampling and importance sampling, have been explained and an introduction to the more advanced methods of Markov Chain Monte Carlo has been provided. We end this chapter by just briefly mentioning some of the most important Monte Carlo methods that have not been covered here.

A variance reduction method called *control variates* is presented [Kalos and Whitlock, 1986, pg. 107]. This method can be advantageous in some cases, but requires the existence of a function correlated to the integrand that can be integrated analytically. Another variance reduction method is *antithetic variates* [Kalos and Whitlock, 1986, pg. 109], which achieves a decrease in variance using random variables that are negatively correlated. Intelligent sample placement is another important challenge. *Adaptive sampling* is a method for placing more samples in high variance regions, the pitfalls of which are examined by Kirk and Arvo [1991]. *Latin hypercube sampling* is a way generating high-dimensional sampling patterns [Shirley, 1991], whereas *quasi-Monte Carlo* uses low-discrepancy sequences to produce well-distributed samples [Keller, 1995].

Chapter 4

Solution Strategies

In this chapter we will discuss methods for solving the light transport problem. We will begin by rephrasing the light transport problem as an integration problem, thus making it suitable for applying the Monte Carlo techniques presented in Chapter 3. To do so, we first rewrite the equation of transfer in the simpler integral form. By combining the integral form with the measurement equation, it is possible to formulate the light transport problem as the integral of a specific function, called the measurement contribution function, over an abstract space of paths.

The path integral formulation has the advantage of stating the light transport problem in a way that is both simpler to understand and which makes it easier to apply more advanced Monte Carlo methods. Using the path integral formulation, we explain the classical unbiased algorithms based on regular Monte Carlo (path tracing, light tracing, and bidirectional path tracing) and also the algorithms based Markov Chain Monte Carlo (e.g. Metropolis light transport).

We conclude the chapter with a discussion of the challenges of importance sampling according to the measurement contribution function.

4.1 Integral Form of Light Transport Problem

As discussed in the previous chapter, the equilibrium distribution of radiance is governed by the equation of transfer,

$$\boldsymbol{\omega} \cdot \nabla L_{\mathrm{o}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}) + \sigma_{\mathrm{t}}(\mathbf{x}_{\mathrm{v}})\, L_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}}, -\boldsymbol{\omega}) = L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}) + \sigma_{\mathrm{s}}(\mathbf{x}_{\mathrm{v}}) \int_{\mathcal{S}^2} f_{\mathrm{p}}(\mathbf{x}_{\mathrm{v}}, -\boldsymbol{\omega}' \cdot \boldsymbol{\omega})\, L_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega}')\, \mathrm{d}\sigma(\boldsymbol{\omega}'),$$

with boundary conditions given by the local scattering equation,

$$L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}) = L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}) + \int_{\mathcal{S}^2} L_{\mathrm{i}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}')\, f_{\mathrm{s}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega}', \boldsymbol{\omega})\, |\cos\theta|\ \mathrm{d}\sigma(\boldsymbol{\omega}'),$$

where $\mathbf{x}_{\mathrm{v}} \in \mathcal{V}_0$ and $\mathbf{x}_{\mathrm{s}} \in \partial\mathcal{V}$ as usual.

Due to the directional derivative, $\boldsymbol{\omega} \cdot \nabla L_{\mathrm{o}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega})$, these equations are not convenient to work with. It turns out to be possible to rewrite the equation of transfer in integral form and at the same time incorporate the boundary conditions. This transforms the original problem from an integro-differential equation to an integral equation, which is easier to manipulate. As shown by Arvo [1993], this can be done by integrating each side of the equation of transfer along a ray segment.

Before presenting the equation of transfer in integral form, it is necessary to define two utility functions. The first is the *boundary distance function*, which returns the distance to the boundary along a ray,

$$d_{\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega}) = \inf\left\{ s > 0 \mid \mathbf{x} + s\,\boldsymbol{\omega} \in \partial\mathcal{V} \right\}. \tag{4.1}$$

We will assume that the scene is closed, so that $d_{\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega}) < \infty$. If the scene is open, this can be achieved by enclosing it in a bounding sphere that is totally absorbing. The nearest point on the boundary along a ray is then given by the *ray-casting function*

$$\mathbf{x}_{\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega}) = \mathbf{x} + d_{\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega})\, \boldsymbol{\omega}, \tag{4.2}$$

where $\mathbf{x}_{\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega}) \in \partial\mathcal{V}$ holds, since $d_{\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega}) < \infty$.

We will rewrite the equation of transfer by integrating each side of the equation along a ray segment, which requires us to compute how radiance is changed as it traverses a participating medium. The decrease in radiance along a line segment depends on the *optical depth* of the medium the ray traverses. The optical depth can be computed by integrating the extinction coefficient along the ray segment,

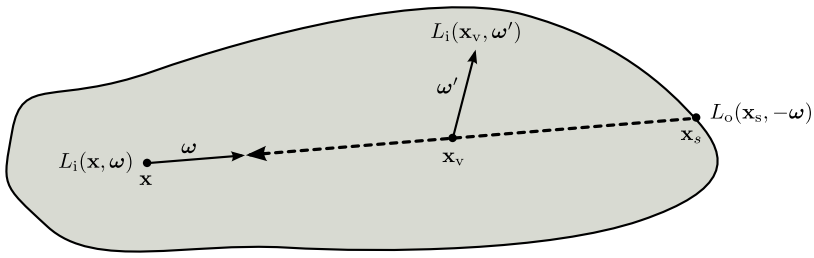
$$\tau(\mathbf{x} \leftrightarrow \mathbf{x}') = \int_0^{\|\mathbf{x}-\mathbf{x}'\|} \sigma_{\mathrm{t}}(\mathbf{x} + s\,\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{x}'})\ \mathrm{d}s. \tag{4.3}$$

This function is symmetric function, as is indicated by the double arrows. The optical depth can then be used to compute the fraction of radiance transmitted from $\mathbf{x}'$ to $\mathbf{x}$ that is not absorbed or outscattered. This quantity is called the *transmittance*, and is given by

$$\mathcal{T}(\mathbf{x}\leftrightarrow\mathbf{x}') = \begin{cases} \exp\Big[-\tau(\mathbf{x}\leftrightarrow\mathbf{x}')\Big] & \text{if } \|\mathbf{x}-\mathbf{x}'\| \le d_{\partial\mathcal{V}}(\mathbf{x},\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{x}'}), \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

The transmittance decays exponentially as a function of distance. If the ray segment from $\mathbf{x}$ to $\mathbf{x}'$ intersects the boundary, the transmittance is zero due to refraction.

Figure 4.1: The equation of transfer in integral form along a ray segment is expressed as the sum of two components. The first is the radiance scattered into the ray at the boundary given by the local scattering equation and suitably adjusted by the transmittance along the ray segment. The second is the integral of the sum of inscattered and emitted radiance along the ray, also attenuated by the transmittance term.



Using the above definitions, the equation of transfer can now be rewritten in pure integral form. Consider the radiance at $\mathbf{x}$ incident from direction $\boldsymbol{\omega}$, $L_{\mathrm{i}}(\mathbf{x},\boldsymbol{\omega})$, as illustrated in Figure 4.1. Let $\mathbf{x}_{\mathrm{s}} = \mathbf{x}_{\partial\mathcal{V}}(\mathbf{x},\boldsymbol{\omega})$ be the point on the boundary along the ray with exitant radiance $L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}},-\boldsymbol{\omega})$ given by the local scattering equation. Then, the equation of transfer can be expressed in integral form as a sum of two components,

$$L_{\mathrm{i}}(\mathbf{x},\boldsymbol{\omega}) = \mathcal{T}(\mathbf{x}\leftrightarrow\mathbf{x}_{\mathrm{s}})\, L_{\mathrm{o}}(\mathbf{x}_{\mathrm{s}},-\boldsymbol{\omega}) + \int_0^{\|\mathbf{x}-\mathbf{x}_{\mathrm{s}}\|} \mathcal{T}(\mathbf{x}\leftrightarrow\mathbf{x}_{\mathrm{v}}) \left[L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_{\mathrm{v}},-\boldsymbol{\omega}) + \sigma_{\mathrm{s}}(\mathbf{x}_{\mathrm{v}}) \int_{\mathcal{S}^2} f_{\mathrm{p}}(\mathbf{x}_{\mathrm{v}},-\boldsymbol{\omega}'\cdot-\boldsymbol{\omega})\, L_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}},\boldsymbol{\omega}')\ \mathrm{d}\sigma(\boldsymbol{\omega}')\right] \mathrm{d}s \tag{4.5}$$

where $\mathbf{x}_{\mathrm{v}} = \mathbf{x} + s\,\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{x}_{\mathrm{s}}}$ are points in the volume along the ray. The first component is the radiance scattered into the ray at the boundary. The second is the integral of the sum of the emitted and inscattered radiance along the ray. Both of these terms must be attenuated by the transmittance term to account for absorption and outscattering.

4.2 Operator Form

The equation of transfer can also be rephrased in terms of linear operators acting on functions from some infinite function space, such as L^p (see Hansen [2006] for a very readable introduction to functional analysis). As discussed in the next sections, this approach has the advantage that it becomes possible to show under what circumstances solutions to the light transport problem exist. It also directly leads to new insight that can be used to construct algorithms for solving the problem.

Operators have also been used in the field of computer graphics to study the light transport problem. In the special case of surface light transport (i.e. no participating media), which is the case that has been studied most, Arvo [1995a,b], Lafortune [1995], Dutre [1996], and Veach [1996, 1997] all use this approach. The general case with participating media is covered in Pauly [1999] and Pauly, Kollig, and Keller [2000]. Our approach is based on Veach [1997], but extended to participating media.

The equation of transfer can be expressed using linear operators by separating out the emission terms ($L_{\mathrm{e},\partial\mathcal{V}}$ and $L_{\mathrm{e},\mathcal{V}_0}$) from the actual light transport process. Light transport can be seen as consisting of two alternating steps that are repeated indefinitely. The first step is scattering, where incident radiance is transformed to exitant radiance using a scattering kernel, which is either the BSDF or phase function. The second step is propagation, where exitant radiance is distributed throughout the scene. Since we distinguish between points on surfaces, $\partial\mathcal{V}$, and points in the volume, $\mathcal{V}_0$, it will be necessary to define a pair scattering and propagation operators for each.

The *surface scattering operator* is given by

$$(\mathbf{K}_{\partial\mathcal{V}}\, h)(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{o}}) \equiv \begin{cases} \displaystyle\int_{\mathcal{S}^2} h(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}})\, f_{\mathrm{s}}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}})\, |\cos\theta|\ \mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{i}}), & \text{if } \mathbf{x} \in \partial\mathcal{V} \\ 0, & \text{if } \mathbf{x} \in \mathcal{V}_0, \end{cases}$$

and transforms incident radiance functions at surfaces to exitant radiance functions. Since BSDFs are not defined for points in the volume, this portion of L_{o} is simply set to zero.

The *volume scattering operator* is given by

$$(\mathbf{K}_{\mathcal{V}_0}\, h)(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{o}}) \equiv \begin{cases} 0, & \text{if } \mathbf{x} \in \partial\mathcal{V}, \\ \sigma_{\mathrm{s}}(\mathbf{x}) \displaystyle\int_{\mathcal{S}^2} h(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}})\, f_{\mathrm{p}}(\mathbf{x}, -\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{o}})\ \mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{i}}), & \text{if } \mathbf{x} \in \mathcal{V}_0, \end{cases}$$

and transforms incident radiance functions in the volume to exitant radiance functions. This operator only applies to points in the volume, i.e. where f_p is defined, and is thus complementary to the surface scattering operator. The operator that describe scattering at all points in $\mathcal{V}$ is then simply the sum of these two operators,

$$\mathbf{K} \equiv \mathbf{K}_{\partial\mathcal{V}} + \mathbf{K}_{\mathcal{V}_0},$$

which is known simply as the *scattering operator*.

The other part of light transport is propagation. The *surface propagation operator* is given by

$$(\mathbf{G}_{\partial\mathcal{V}} h)(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}) \equiv \mathcal{T}(\mathbf{x} \leftrightarrow \mathbf{x}_\mathrm{s})\, h(\mathbf{x}_\mathrm{s}, -\boldsymbol{\omega}_\mathrm{i}),$$

with $\mathbf{x}_\mathrm{s} = \mathbf{x}_{\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})$. This operator propagates exitant radiance from surfaces. Similarly, the *volume propagation operator* transforms exitant radiance functions in the volume into incident radiance functions,

$$(\mathbf{G}_{\mathcal{V}_0} h)(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}) \equiv \int_0^{\|\mathbf{x}-\mathbf{x}_\mathrm{s}\|} \mathcal{T}(\mathbf{x} \leftrightarrow \mathbf{x}_\mathrm{v})\, h(\mathbf{x}_\mathrm{v}, -\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}s,$$

where $\mathbf{x}_\mathrm{v} = \mathbf{x} + s\,\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{x}_\mathrm{s}}$ ($L_\mathrm{i} = \mathbf{G}_{\mathcal{V}_0} L_\mathrm{o}$). Like the scattering operator, it is convenient to define a single operator that accounts for both types of propagation. This operator is called the *propagation operator* and is simply the sum of the previous two operators,

$$\mathbf{G} \equiv \mathbf{G}_{\partial\mathcal{V}} + \mathbf{G}_{\mathcal{V}_0}.$$

The scattering and propagation operators can be combined in two ways to form the *light transport operators*,

$$\mathbf{T}_{L_\mathrm{o}} \equiv \mathbf{K}\,\mathbf{G} \quad \text{and} \quad \mathbf{T}_{L_\mathrm{i}} \equiv \mathbf{G}\,\mathbf{K}.$$

The reason that there are two light transport operators is that it is possible to express the equilibrium distribution of radiance in both exitant and incident form.

Using the source terms and the transport operators, the complete equation of transfer including boundary conditions can be expressed in terms of exitant radiance as

$$L_\mathrm{o} = (L_{\mathrm{e},\partial\mathcal{V}} + L_{\mathrm{e},\mathcal{V}_0}) + \mathbf{T}_{L_\mathrm{o}}\, L_\mathrm{o}. \tag{4.6}$$

Alternatively, the equilibrium distribution can be expressed in terms of incident radiance,

$$L_\mathrm{i} = \mathbf{G}(L_{\mathrm{e},\partial\mathcal{V}} + L_{\mathrm{e},\mathcal{V}_0}) + \mathbf{T}_{L_\mathrm{i}}\, L_\mathrm{i}. \tag{4.7}$$

In the latter case, it is necessary to transform the source terms, which are given as part of the description in exitant form, to incident form using a single application of the propagation operator. Both of these equations are examples of Fredholm integral equations of the second kind.

4.3 Formal Solution

A formal solution to the light transport problem can be found by first isolating L_o in Equation 4.6 and then inserting this quantity into the measurement equation. Alternatively, L_i could be isolated in Equation 4.7. However, the exitant radiance case is the most commonly used, so in the following we will use this case and drop the subscript on $\mathbf{T}_{L_\mathrm{o}}$. The exitant radiance can be isolated as

$$\begin{aligned} L_\mathrm{o} - \mathbf{T}\,L_\mathrm{o} &= (L_{\mathrm{e},\partial\mathcal{V}} + L_{\mathrm{e},\mathcal{V}_0}) \\ (\mathbf{I} - \mathbf{T})\,L_\mathrm{o} &= (L_{\mathrm{e},\partial\mathcal{V}} + L_{\mathrm{e},\mathcal{V}_0}) \\ L_\mathrm{o} &= (\mathbf{I} - \mathbf{T})^{-1}(L_{\mathrm{e},\partial\mathcal{V}} + L_{\mathrm{e},\mathcal{V}_0}), \end{aligned} \tag{4.8}$$

where $\mathbf{I}$ is the identity operator ($\mathbf{I}L = L$).

The inverse of the operator in the last line of Equation 4.8 is guaranteed to exist if $\|\mathbf{T}\| < 1$, where $\|\mathbf{T}\|$ is the operator norm. If this is the case, this operator, which is known as the solution operator, can be written as a Neumann series [Hansen, 2006],

$$\mathbf{S} = (\mathbf{I} - \mathbf{T})^{-1} = \mathbf{I} + \mathbf{T} + \mathbf{T}^2 + \mathbf{T}^3 + \cdots = \sum_{i=0}^{\infty} \mathbf{T}^i. \tag{4.9}$$

The solution operator also exists in some cases when $\|\mathbf{T}\| \geq 1$, which can happen even for physically valid scenes (i.e. scenes where scattering always results in some absorption). However, the inverse operator exists as long as the series in Equation 4.9 is convergent, even when $\|\mathbf{T}\| \geq 1$.

If Equation 4.9 is substituted back into Equation 4.8, the equilibrium radiance can be written as

$$L_\mathrm{o} = L_\mathrm{e} + \mathbf{T}\,L_\mathrm{e} + \mathbf{T}^2\,L_\mathrm{e} + \mathbf{T}^3\,L_\mathrm{e} + \cdots = \sum_{i=0}^{\infty} \mathbf{T}^i\,L_\mathrm{e} = \mathbf{S}\,L_\mathrm{e},$$

where we used the shorthand $L_\mathrm{e} = L_{\mathrm{e},\partial\mathcal{V}} + L_{\mathrm{e},\mathcal{V}_0}$. This means that equilibrium distribution of exitant radiance in a scene is the sum of emitted radiance and emitted radiance transported once, twice, thrice, and so forth.

Once we know the distribution of radiance, we can compute a measurement by evaluating the measurement equation. Measurements are typically evaluated by computing the inner product of the flux responsivity function (an exitant quantity) with the incident radiance. This can be done by noting that $L_\mathrm{i} = \mathbf{G}\,\mathbf{S}\,L_\mathrm{e}$, so that

$$\begin{aligned} I &= \int_{\mathcal{S}^2}\int_{\mathcal{V}} L_\mathrm{i}(\mathbf{x},\boldsymbol{\omega})\; W_\mathrm{e}(\mathbf{x},\boldsymbol{\omega})\;\, \mathrm{d}\alpha_{\boldsymbol{\omega}}(\mathbf{x})\;\, \mathrm{d}\sigma(\boldsymbol{\omega}) \\ &= \langle L_\mathrm{i}, W_\mathrm{e}\rangle = \langle \mathbf{G}\,L_\mathrm{o}, W_\mathrm{e}\rangle = \langle \mathbf{G}\,\mathbf{S}\,L_\mathrm{e}, W_\mathrm{e}\rangle, \end{aligned} \tag{4.10}$$

where we have used bracket notation to denote inner product. We have also simplified notation a bit by dropping the j superscript on W_{e}^{j}.

4.4 Adjoint Operators

The adjoint of an operator $\mathbf{O}$ is an operator, denoted $\mathbf{O}^*$, which satisfies

$$\langle \mathbf{O}\, F, G \rangle = \langle F, \mathbf{O}^* G \rangle,$$

for any functions F and G. Adjoint operators can be used to gain additional insight into the light transport problem by investigating the adjoints of the operators defined thus far. In particular, if adjoint operators are used in Equation 4.10, we get

$$I = \langle \mathbf{G}\, \mathbf{S}_{L_{\mathrm{o}}}\, L_{\mathrm{e}}, W_{\mathrm{e}} \rangle = \langle L_{\mathrm{e}}, (\mathbf{G}\, \mathbf{S}_{L_{\mathrm{o}}})^*\, W_{\mathrm{e}} \rangle.$$

What this means is that the light transport problem can also be solved by considering how some quantity, that is emitted from the sensors, is transported through the scene and impinges on the light sources. Scattering and propagation of this quantity, which is called importance or potential, does not correspond to any physical process, but is rather a side effect of an inherent symmetry between sensors and light sources present in the mathematical model.

The name "importance" comes from the fact that this quantity tells us which parts of the scene influence the solution the most. This means that importance can be used as a guide to concentrate global illumination calculations where they matter most. As discussed in Christensen [2003], importance is defined as a specific adjoint of radiance. Integral equations, such as Equation 4.6 or Equation 4.7, have an infinite number of adjoint equations each with a different source term. One of these adjoint equations will have a source term (W_{e}) that defines which parts of the function domain that matter most for solving a particular problem, and it is the solution to this equation that is called the importance.

Like radiance, importance exists in both exitant and incident versions, W_{o} and W_{i}. However, the rules that govern how importance is scattered and propagated are different from radiance and are instead given by the adjoints of the previously defined operators.

The adjoint operators in the case with no participating media are covered by Veach [1997]. They show that the surface propagation operator is self-adjoint, i.e. $\mathbf{G}_{\partial\mathcal{V}} = \mathbf{G}_{\partial\mathcal{V}}^*$. They also show that the surface scattering operator is self-adjoint only if the BSDF is symmetric. However, as discussed in Chapter 2,

BSDFs are generally not symmetric, and therefore it is necessary to define the adjoint surface scattering operator, $\mathbf{K}^*_{\partial\mathcal{V}}$, given by

$$(\mathbf{K}^*_{\partial\mathcal{V}}\, L_\mathrm{i})(\mathbf{x},\boldsymbol{\omega}_\mathrm{o}) \equiv \begin{cases} \displaystyle\int_{\mathcal{S}^2} L_\mathrm{i}(\mathbf{x},\boldsymbol{\omega}_\mathrm{i})\, f^*_\mathrm{s}(\mathbf{x},\boldsymbol{\omega}_\mathrm{i},\boldsymbol{\omega}_\mathrm{o})\, |\cos\theta|\ \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}), & \text{if } \mathbf{x}\in\partial\mathcal{V} \\ 0, & \text{if } \mathbf{x}\in\mathcal{V}_0. \end{cases}$$

The only difference from the ordinary surface scattering operator is the use of the adjoint BSDF. See Veach [1997, pg. 130] for proofs.

The proofs for the volume operators are similar. Since the phase function is symmetric, the volume scattering operator is self-adjoint, i.e. $\mathbf{K}^*_{\mathcal{V}_0} = \mathbf{K}_{\mathcal{V}_0}$. The adjoint scattering operator is then simply

$$\mathbf{K}^* = \mathbf{K}^*_{\partial\mathcal{V}} + \mathbf{K}^*_{\mathcal{V}_0}.$$

The volume propagation operator is also self-adjoint, so $\mathbf{G}^*_{\mathcal{V}_0} = \mathbf{G}_{\mathcal{V}_0}$.

Based on the adjoint scattering operator and the propagation operator, the *importance transport operators* can be defined by

$$\mathbf{T}_{W_\mathrm{o}} \equiv \mathbf{K}^*\,\mathbf{G} \quad \text{and} \quad \mathbf{T}_{W_\mathrm{i}} \equiv \mathbf{G}\,\mathbf{K}^*.$$

Using the importance transport operators and the radiance transport operators, the four equations governing the equilibrium distribution of exitant/incident radiance/importance can be written

$$L_\mathrm{o} = L_\mathrm{e} + \mathbf{T}_{L_\mathrm{o}}\, L_\mathrm{o}, \qquad L_\mathrm{i} = \mathbf{G}\, L_\mathrm{e} + \mathbf{T}_{L_\mathrm{i}}\, L_\mathrm{i},$$

$$W_\mathrm{o} = W_\mathrm{e} + \mathbf{T}_{W_\mathrm{o}}\, W_\mathrm{o}, \quad W_\mathrm{i} = \mathbf{G}\, W_\mathrm{e} + \mathbf{T}_{W_\mathrm{i}}\, W_\mathrm{i}.$$

The corresponding solution operators for the equilibrium distribution of exitant and incident importance are given by

$$\mathbf{S}_{W_\mathrm{o}} = (\mathbf{I} - \mathbf{T}_{W_\mathrm{o}})^{-1} \quad \text{and} \quad \mathbf{S}_{W_\mathrm{i}} = (\mathbf{I} - \mathbf{T}_{W_\mathrm{i}})^{-1},$$

whereas the solution operators for exitant and incident radiance were given by

$$\mathbf{S}_{L_\mathrm{o}} = (\mathbf{I} - \mathbf{T}_{L_\mathrm{o}})^{-1} \quad \text{and} \quad \mathbf{S}_{L_\mathrm{i}} = (\mathbf{I} - \mathbf{T}_{L_\mathrm{i}})^{-1}.$$

This leads to the four ways of evaluating the measurement equation,

$$I = \langle \mathbf{G}\,\mathbf{S}_{L_\mathrm{o}}\, L_\mathrm{e}, W_\mathrm{e}\rangle = \langle \mathbf{S}_{L_\mathrm{i}}\,\mathbf{G}\, L_\mathrm{e}, W_\mathrm{e}\rangle = \langle L_\mathrm{e}, \mathbf{G}\,\mathbf{S}_{W_\mathrm{o}}\, W_\mathrm{e}\rangle = \langle L_\mathrm{e}, \mathbf{S}_{W_\mathrm{i}}\,\mathbf{G}\, W_\mathrm{e}\rangle.$$

4.5 Path Integral Formulation

The path integral formulation of light transport was first introduced by Veach and Guibas [1994] and Veach [1997] and later extended to participating media by Pauly [1999] and Pauly et al. [2000].

The idea is to express the light transport problem as a simple integral,

$$I = \int_{\Omega} f(\bar{\mathbf{x}}) \, \mathrm{d}\mu(\bar{\mathbf{x}}),$$

of a function called the measurement contribution function over an abstract space of paths. The advantage of this formulation is simplicity. So far, the light transport problem has been expressed as the inner product of a flux responsivity function with the incident radiance (the measurement equation), where the incident radiance is only defined implicitly through a fairly complicated integral equation (the equation of transfer in integral form). Instead, with the path integral formulation, the light transport problem is expressed as an ordinary integral of a simple function over the paths connecting sensors and light sources.

4.5.1 Path Space and Path Space Measure

The path integral formulation is based on rewriting the measurement equation as an integral over path space. To do so, it is necessary to define what exactly is meant by path, path space, and what the associated measure is. In the following we use the definition given by Pauly [1999].

A path of length $k \geq 1$ is defined as a sequence of $k+1$ vertices,

$$\bar{\mathbf{x}} = \mathbf{x}_0 \, \mathbf{x}_1 \ldots \mathbf{x}_k, \tag{4.11}$$

where the vertices can be either on the surfaces of scene, $\partial\mathcal{V}$, or in the volume these surfaces enclose, $\mathcal{V}_0$.

To define the notion of path space, it is necessary to consider all the ways paths can be formed. A given path of length k can be formed in 2^{k+1} ways, since each vertex can belong to either $\partial\mathcal{V}$ or $\mathcal{V}_0$. The $k+1$ bit representation of this relationship is called the *path characteristic*, denoted $l \in \mathbb{N}$, of the path. To make this more concrete, let $b_i(l)$ be the ith bit of the binary representation of l. Then, we will define $b_i(l)$ to be 1 if $\mathbf{x}_i \in \partial\mathcal{V}$ and 0 if $\mathbf{x}_i \in \mathcal{V}_0$.

Using the above definitions, the space of paths of length k, with path characteristic l is defined by

$$\Omega_k^l \equiv \left\{ \bar{\mathbf{x}} = \mathbf{x}_0\, \mathbf{x}_1 \ldots \mathbf{x}_k \;\middle|\; \mathbf{x}_i \in \begin{cases} \partial\mathcal{V} & \text{if } b_i(l) = 1 \\ \mathcal{V}_0 & \text{if } b_i(l) = 0 \end{cases} \right\}$$

with $0 \leq l < 2^{k+1}$.

A measure on Ω_k^l can be defined as

$$\mu_k^l(A) = \int_A \prod_{i=0}^{k} \mathrm{d}\mu_{k,i}^l(\bar{\mathbf{x}})$$

where $A \subseteq \Omega_k^l$. The differential measure, $\mathrm{d}\mu_{k,i}^l(\bar{\mathbf{x}})$, is given by

$$\mathrm{d}\mu_{k,i}^l(\mathbf{x}_0\, \mathbf{x}_1 \ldots \mathbf{x}_k) \equiv \begin{cases} \mathrm{d}A(\mathbf{x}_i) & \text{if } b_i(l) = 1, \\ \mathrm{d}V(\mathbf{x}_i) & \text{if } b_i(l) = 0. \end{cases}$$

The space of paths of length k with any path characteristic, $0 \leq l < 2^{k+1}$, is given as the union,

$$\Omega_k \equiv \bigcup_{l=0}^{2^{k+1}-1} \Omega_k^l,$$

with associated measure

$$\mu_k(B) = \sum_{l=0}^{2^{k+1}-1} \mu_k^l(B \cap \Omega_k^l),$$

where $B \subseteq \Omega_k$. Finally, *path space* is defined as the union of path spaces containing paths of all lengths,

$$\Omega \equiv \bigcup_{k=1}^{\infty} \Omega_k.$$

with associated measure

$$\mu(C) = \sum_{k=1}^{\infty} \mu_k(C \cap \Omega_k),$$

and where $C \subseteq \Omega$.

4.5.2 Three Point Form of the Equation of Transfer

The equation of transfer, presented in Section 4.1, is a set of equations that define the equilibrium radiance in a scene. In particular, for points $\mathbf{x}_\mathrm{v} \in \mathcal{V}_0$, radiance is determined by volume scattering and emission,

$$L_\mathrm{s}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) = L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}) + \sigma_\mathrm{s}(\mathbf{x}_\mathrm{v}) \int_{\mathcal{S}^2} f_\mathrm{p}(\mathbf{x}_\mathrm{v}, -\boldsymbol{\omega}' \cdot \boldsymbol{\omega})\, L_\mathrm{i}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}')\, \mathrm{d}\sigma(\boldsymbol{\omega}'),$$

and for points $\mathbf{x}_\mathrm{s} \in \partial\mathcal{V}$, radiance is determined by surface scattering and emission,

$$L_\mathrm{o}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}) = L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}) + \int_{\mathcal{S}^2} f_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}, \boldsymbol{\omega}')\, L_\mathrm{i}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}')\, |\cos\theta|\, \mathrm{d}\sigma(\boldsymbol{\omega}').$$

Exitant radiance is related to incident radiance by integrating along a ray,

$$L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}) = \mathcal{T}(\mathbf{x} \leftrightarrow \mathbf{x}_\mathrm{s})\, L_\mathrm{o}(\mathbf{x}_\mathrm{s}, -\boldsymbol{\omega}) + \int_0^{\|\mathbf{x}-\mathbf{x}_\mathrm{s}\|} \mathcal{T}(\mathbf{x} \leftrightarrow \mathbf{x}_\mathrm{v})\, L_\mathrm{s}(\mathbf{x}_\mathrm{v}, -\boldsymbol{\omega})\, \mathrm{d}s,$$

where $\mathbf{x}_\mathrm{s} = \mathbf{x}_{\partial\mathcal{V}}(\mathbf{x}, \boldsymbol{\omega})$ and $\mathbf{x}_\mathrm{v} = \mathbf{x} + s\,\boldsymbol{\omega}$. This is simply the equation of transfer in integral form, where the terms involving volume and surface interaction have been separated out.

In the above form, the equation of transfer is expressed using integrals over directions in $\mathcal{S}^2$. In order to use these equations with the path integral formulation, it will be necessary to express these directions in terms of path vertices, rather than directions. This leads to the three point form of the equation of transfer.

We will begin by defining incident and exitant radiance in terms of path vertices,

$$\begin{aligned} L(\mathbf{x}_1 \leftarrow \mathbf{x}_2) &= L_\mathrm{i}(\mathbf{x}_1, \boldsymbol{\omega}_{\mathbf{x}_1 \rightarrow \mathbf{x}_2}), \\ L(\mathbf{x}_1 \rightarrow \mathbf{x}_2) &= L_\mathrm{o}(\mathbf{x}_1, \boldsymbol{\omega}_{\mathbf{x}_1 \rightarrow \mathbf{x}_2}) \end{aligned}$$

where it is assumed that $\mathbf{x}_1 \neq \mathbf{x}_2$. This representation has some redundancy, since $L(\mathbf{x}_1 \leftarrow \mathbf{x}_2) = L(\mathbf{x}_1 \leftarrow \mathbf{x}_3)$ whenever $\boldsymbol{\omega}_{\mathbf{x}_1 \rightarrow \mathbf{x}_2} = \boldsymbol{\omega}_{\mathbf{x}_1 \rightarrow \mathbf{x}_3}$.

Using the new notation, volume interaction can be expressed as

$$\begin{aligned} L(\mathbf{x}_\mathrm{v} \rightarrow \mathbf{x}_\mathrm{o}) = L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_\mathrm{v} \rightarrow \mathbf{x}_\mathrm{o}) + \\ \sigma_\mathrm{s}(\mathbf{x}_\mathrm{v}) \int_{\mathcal{V}} f_\mathrm{p}(\mathbf{x}_\mathrm{i} \rightarrow \mathbf{x}_\mathrm{v} \rightarrow \mathbf{x}_\mathrm{o}) L(\mathbf{x}_\mathrm{i} \rightarrow \mathbf{x}_\mathrm{v})\, G(\mathbf{x}_\mathrm{i} \leftrightarrow \mathbf{x}_\mathrm{v})\, \mathrm{d}\lambda(\mathbf{x}_\mathrm{i}), \end{aligned} \tag{4.12}$$

and surface interaction can be expressed as

$$L(\mathbf{x}_\mathrm{s}\to\mathbf{x}_\mathrm{o}) = L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_\mathrm{s}\to\mathbf{x}_\mathrm{o}) + \int_{\mathcal{V}} f_\mathrm{s}(\mathbf{x}_\mathrm{i}\to\mathbf{x}_\mathrm{s}\to\mathbf{x}_\mathrm{o})\, L(\mathbf{x}_\mathrm{i}\to\mathbf{x}_\mathrm{s})\, G(\mathbf{x}_\mathrm{i}\leftrightarrow\mathbf{x}_\mathrm{s})\ \mathrm{d}\lambda(\mathbf{x}_\mathrm{i}). \tag{4.13}$$

Here we have just changed the domain of integration from $\mathcal{S}^2$ to $\mathcal{V}$. Since the integration is over exitant radiance at all points, there is no need for a third equation to describe the relationship between exitant radiance and incident radiance.

The BSDF and phase function are defined using the new notation as

$$\begin{aligned} f_\mathrm{s}(\mathbf{x}_\mathrm{i}\to\mathbf{x}_\mathrm{s}\to\mathbf{x}_\mathrm{o}) &= f_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}_{\mathbf{x}_\mathrm{s}\to\mathbf{x}_\mathrm{i}}, \boldsymbol{\omega}_{\mathbf{x}_\mathrm{s}\to\mathbf{x}_\mathrm{o}}), \text{ and} \\ f_\mathrm{p}(\mathbf{x}_\mathrm{i}\to\mathbf{x}_\mathrm{v}\to\mathbf{x}_\mathrm{o}) &= f_\mathrm{p}(\mathbf{x}_\mathrm{v}, \boldsymbol{\omega}_{\mathbf{x}_\mathrm{s}\to\mathbf{x}_\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathbf{x}_\mathrm{s}\to\mathbf{x}_\mathrm{o}}). \end{aligned}$$

The function G is called the *geometry term*. It is necessary to introduce this term to account for the change of variables from the solid angle measure to the area/volume measure. The ordinary geometry term is given by

$$G_{\partial\mathcal{V}}(\mathbf{x}_1\leftrightarrow\mathbf{x}_2) = \mathcal{T}(\mathbf{x}_1\leftrightarrow\mathbf{x}_2)\frac{|\mathbf{n}_\mathrm{g}(\mathbf{x}_1)\cdot\boldsymbol{\omega}_{\mathbf{x}_1\to\mathbf{x}_2}|\,|\mathbf{n}_\mathrm{g}(\mathbf{x}_2)\cdot\boldsymbol{\omega}_{\mathbf{x}_2\to\mathbf{x}_1}|}{\|\mathbf{x}_1-\mathbf{x}_2\|^2}.$$

The ordinary geometry term is only defined when $\mathbf{x}_1, \mathbf{x}_2 \in \partial\mathcal{V}$, since it depends on the normals. To also account for points in the volume, the *generalized geometry term* presented by Pauly et al. [2000] can be used instead. First, let

$$g(\mathbf{x}\to\mathbf{x}') = \begin{cases} |\mathbf{n}_\mathrm{g}(\mathbf{x})\cdot\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{x}'}| & \text{if } \mathbf{x}\in\partial\mathcal{V} \\ 1 & \text{otherwise.} \end{cases}$$

Then the generalized geometry term is given by

$$G(\mathbf{x}_1\leftrightarrow\mathbf{x}_2) = \mathcal{T}(\mathbf{x}_1\leftrightarrow\mathbf{x}_2)\,\frac{g(\mathbf{x}_1\to\mathbf{x}_2)\,g(\mathbf{x}_2\to\mathbf{x}_1)}{\|\mathbf{x}_1-\mathbf{x}_2\|^2}.$$

Equations 4.12 and 4.13, which describe scattering and emission in the volume and at surfaces, have similar form. It turns out to be convenient to be able to describe scattering and emission independently of whether the vertex belongs to $\mathcal{V}_0$ or $\partial\mathcal{V}$. To this end we define the *scattering kernel* as

$$f_\mathrm{k}(\mathbf{x}_\mathrm{i}\to\mathbf{x}\to\mathbf{x}_\mathrm{o}) = \begin{cases} f_\mathrm{s}(\mathbf{x}_\mathrm{i}\to\mathbf{x}\to\mathbf{x}_\mathrm{o}) & \text{if } \mathbf{x}\in\partial\mathcal{V} \\ \sigma_\mathrm{s}(\mathbf{x})\, f_\mathrm{p}(\mathbf{x}_\mathrm{i}\to\mathbf{x}\to\mathbf{x}_\mathrm{o}) & \text{if } \mathbf{x}\in\mathcal{V}_0, \end{cases}$$

and the *adjoint scattering kernel* as

$$f^*_\mathrm{k}(\mathbf{x}_\mathrm{i}\to\mathbf{x}\to\mathbf{x}_\mathrm{o}) = \begin{cases} f^*_\mathrm{s}(\mathbf{x}_\mathrm{i}\to\mathbf{x}\to\mathbf{x}_\mathrm{o}) & \text{if } \mathbf{x}\in\partial\mathcal{V} \\ \sigma_\mathrm{s}(\mathbf{x})\, f_\mathrm{p}(\mathbf{x}_\mathrm{i}\to\mathbf{x}\to\mathbf{x}_\mathrm{o}) & \text{if } \mathbf{x}\in\mathcal{V}_0. \end{cases}$$

Note this is a slight abuse of notation, since the units differ (the BSDF and phase function both have units $1/\mathrm{sr}$, but the scattering coefficient has units $1/\mathrm{m}$). However, it simplifies notation, so we will ignore this inconsistency.

Similarly, a function can be defined that describes emission regardless of whether $\mathbf{x} \in \mathcal{V}_0$ or $\mathbf{x} \in \partial\mathcal{V}$. This emission function is given by

$$L_\mathrm{e}(\mathbf{x}_1 \to \mathbf{x}_2) = \begin{cases} L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_1 \to \mathbf{x}_2) & \text{if } \mathbf{x}_1 \in \partial\mathcal{V} \\ L_{\mathrm{e},\mathcal{V}_0}(\mathbf{x}_1 \to \mathbf{x}_2) & \text{if } \mathbf{x}_1 \in \mathcal{V}_0. \end{cases}$$

Here too the units differ (ordinary radiance for surface emission versus volume radiance for volume emission), but we also ignore this for the sake of notational convenience. The importance emission function, W_e, can also be expressed in this way. Let $W_\mathrm{e}(\mathbf{x}_1 \to \mathbf{x}_2) = W_\mathrm{e}(\mathbf{x}_2, \boldsymbol{\omega}_{\mathbf{x}_2 \to \mathbf{x}_1})$, where the arrow notation is still used to indicate the direction of radiance flow (which is the opposite direction of importance flow).

Using these conventions, the equation of transfer that govern the equilibrium distribution of exitant radiance can be expressed as

$$L(\mathbf{x} \to \mathbf{x}_\mathrm{o}) = L_\mathrm{e}(\mathbf{x} \to \mathbf{x}_\mathrm{o}) + \int_\mathcal{V} f_\mathrm{k}(\mathbf{x}_\mathrm{i} \to \mathbf{x} \to \mathbf{x}_\mathrm{o})\, L(\mathbf{x}_\mathrm{i} \to \mathbf{x})\, G(\mathbf{x}_\mathrm{i} \leftrightarrow \mathbf{x})\ \mathrm{d}\lambda(\mathbf{x}_\mathrm{i}). \tag{4.14}$$

Similarly, the distribution of exitant importance can be expressed as

$$W(\mathbf{x}_\mathrm{o} \to \mathbf{x}) = W_\mathrm{e}(\mathbf{x}_\mathrm{o} \to \mathbf{x}) + \int_\mathcal{V} f^*_\mathrm{k}(\mathbf{x}_\mathrm{i} \to \mathbf{x} \to \mathbf{x}_\mathrm{o})\, W(\mathbf{x} \to \mathbf{x}_\mathrm{i})\, G(\mathbf{x} \leftrightarrow \mathbf{x}_\mathrm{i})\ \mathrm{d}\lambda(\mathbf{x}_\mathrm{i}). \tag{4.15}$$

4.5.3 The Measurement Contribution Function

We can now rewrite the measurement equation using the new three point form of the equation of transfer. Assume we have solved the equation of transfer (or its adjoint version) and obtained the equilibrium radiance or importance in the exitant version. In that case, the measurement equation can be written as integral over all paths from Ω_1, i.e. over paths of length 1,

$$\begin{aligned} I &= \int_{\Omega_1} L(\mathbf{x}_0 \to \mathbf{x}_1)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, W_\mathrm{e}(\mathbf{x}_0 \to \mathbf{x}_1)\ \mathrm{d}\mu(\mathbf{x}_0\, \mathbf{x}_1) \\ &= \int_{\Omega_1} W(\mathbf{x}_0 \to \mathbf{x}_1)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, L_\mathrm{e}(\mathbf{x}_0 \to \mathbf{x}_1)\ \mathrm{d}\mu(\mathbf{x}_0\, \mathbf{x}_1). \end{aligned}$$

If we choose to solve for the equilibrium radiance, L can be computed using the Neumann series from Section 4.3. The idea is to recursively expand Equation 4.14,

$$\begin{aligned} I = &\int_{\Omega_1} L_\mathrm{e}(\mathbf{x}_0 \to \mathbf{x}_1)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, W_\mathrm{e}(\mathbf{x}_0 \to \mathbf{x}_1)\ \mathrm{d}\mu(\mathbf{x}_0\mathbf{x}_1) \\ &+ \int_{\Omega_2} L_\mathrm{e}(\mathbf{x}_0 \to \mathbf{x}_1)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, f_\mathrm{k}(\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2)\, G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \\ &\qquad W_\mathrm{e}(\mathbf{x}_1 \to \mathbf{x}_2)\ \mathrm{d}\mu(\mathbf{x}_0\mathbf{x}_1\mathbf{x}_2) \\ &+ \ldots \end{aligned}$$

If the repeated terms are collected, this can also be written

$$\begin{aligned} I = \sum_{k=1}^{\infty} \int_{\Omega_k} & L_\mathrm{e}(\mathbf{x}_0 \to \mathbf{x}_1)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \\ & \left[\prod_{i=1}^{k-1} f_\mathrm{k}(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1})\, G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \right] \\ & W_\mathrm{e}(\mathbf{x}_{k-1} \to \mathbf{x}_k)\ \mathrm{d}\mu_k(\mathbf{x}_0\, \mathbf{x}_1 \ldots \mathbf{x}_k). \end{aligned} \tag{4.16}$$

Alternatively, the equilibrium importance function W could have been expanded. The only difference would have been that the adjoint scattering kernel, f_k^*, should have been used instead, but with the arguments reversed. However, because of the way the adjoint BSDF is defined, this would result in the same numerical value for the measurement, as it should.

The integrand of Equation 4.16 is called the *measurement contribution function*,

$$\begin{aligned} f(\bar{\mathbf{x}}) = & L_\mathrm{e}(\mathbf{x}_0 \to \mathbf{x}_1)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \\ & \left[\prod_{i=1}^{k-1} f_\mathrm{k}(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1})\, G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \right] W_\mathrm{e}(\mathbf{x}_{k-1} \to \mathbf{x}_k). \end{aligned}$$

Using this function, the path integral formulation of the light transport problem can be written simply as

$$I = \int_{\Omega} f(\bar{\mathbf{x}})\ \mathrm{d}\mu(\bar{\mathbf{x}}). \tag{4.17}$$

This makes it clear that the light transport problem is really an integration problem suitable for solving using the Monte Carlo methods described in Chapter 3.

4.6 Sampling Strategies

As shown in Section 3.3, the solution to integration problems of the kind described by Equation 4.17 can be found using estimators of the form

$$I \approx \frac{f(\bar{\mathbf{x}})}{p(\bar{\mathbf{x}})}.$$

It was also shown that such an estimator forms unbiased estimates of I as long $p(\bar{\mathbf{x}})$ is a valid PDF in a Monte Carlo sense. I.e., if $p(\bar{\mathbf{x}}) > 0$ whenever $f(\bar{\mathbf{x}}) \neq 0$, then

$$\mathrm{E}\left[\frac{f(\bar{\mathbf{x}})}{p(\bar{\mathbf{x}})}\right] = I.$$

In order to apply this estimator, a method must exist for generating random paths and for computing the probabilities that each of these paths were generated with.

Finding good methods for generating paths turns out to be a surprisingly difficult problem, and, as we will see, this is the key problem much of the published literature on efficient Monte Carlo ray tracing is concerned with. Recall that we are interested in finding high efficiency estimators, i.e. estimators with low variance that can still generate samples quickly. This means that we should attempt to generate paths distributed according to the measurement contribution function, $f(\bar{\mathbf{x}})$, since this results in low variance estimators according to the principle of importance sampling. Unfortunately, as discussed later in this chapter, generating samples distributed according to $f(\bar{\mathbf{x}})$ is not possible, and consequently compromises have to be made.

Most methods for constructing paths are based on *local path sampling*. Local path sampling is essentially a recipe for constructing paths and computing path probabilities. Using local path sampling directly leads to the three principal ways of constructing paths: namely, constructing path by starting from the sensor (path tracing), constructing paths by starting from the light sources (light tracing), and constructing paths by starting at both sensor and light sources (bidirectional path tracing).

4.6.1 Local Path Sampling

Local path sampling, as described by Veach [1997, pg. 226] for the case with no participating media, is based on constructing paths using three rules. The three rules for constructing paths using random walks in the presence of participating media are

1. The initial vertex in a subpath can be chosen according to some predefined density, $p(\mathbf{x}_0)$, over all of $\mathcal{V}$. This means that these vertices can be sampled from both $\mathcal{V}_0$ and $\partial\mathcal{V}$.

2. Given a vertex in a subpath, $\mathbf{x}_i$, a new vertex can be appended to this subpath. The new vertex is sampled using a conditional density, $p(\mathbf{x}_{i+1}|\mathbf{x}_i)$. This is usually done in two steps. First a random direction is sampled and then $\mathbf{x}_{i+1}$ is found by sampling a random distance along this direction.

3. Finally, vertices from two or more subpaths can be connected to form new paths.

Sampling the Initial Vertex

According to rule 1, the initial vertex in a subpath is sampled using some a priori probability distribution. It is common to chose the initial vertex at a point where $L_\mathrm{e} > 0$ or $W_\mathrm{e} > 0$, though this is not a requirement. For instance, for $\mathbf{x}_0 \in \partial\mathcal{V}$, it is common to make $p(\mathbf{x}_0)$ proportional to radiant exitance.

Once the initial vertex in a subpath has been found, it is necessary to determine the surrounding media. Specifically, for path vertices in $\mathcal{V}_0$ the surrounding medium must be known, whereas for vertices in $\partial\mathcal{V}$ the media on both sides of the boundary must be determined. As discussed in Section 2.6, each cell with a medium can be assigned a precedence value, so that the current medium can always be unambiguously decided in case cells overlap. To facilitate this, each ray can keep a list with all the cells it is currently traversing sorted by precedence. This list can be initialized by tracing a ray from a random position outside $\mathcal{V}$ to the vertex while keeping track of all the cells that are entered and exited along the way.

Sampling Additional Vertices

The second rule states that a vertex can be added to an existing subpath by sampling a new vertex conditional on the last vertex in the subpath. This is done by first sampling a random direction according to some density $p(\boldsymbol{\omega})$, usually expressed with respect to solid angle, and then sampling a distance along this direction using some 1D density, $p(s)$.

The probability of sampling the vertex with respect to the area/volume measure

can be computed as

$$p'(\mathbf{x}_{i+1}|\mathbf{x}_i) = p(\boldsymbol{\omega})\,\frac{g(\mathbf{x}_{i+1}\rightarrow\mathbf{x}_i)}{\|\mathbf{x}_{i+1}-\mathbf{x}_i\|^2}$$

where $\mathbf{x}_{i+1}$ is a point visible along the direction $\boldsymbol{\omega}$ from $\mathbf{x}_i$. If $\mathbf{x}_{i+1}$ cannot be assumed to visible from $\mathbf{x}_i$, a visibility term, $V(\mathbf{x}_i\leftrightarrow\mathbf{x}_{i+1})$, needs to be included in the conversion.

The densities, $p(\boldsymbol{\omega})$, used for sampling directions depend on the vertex. For the first vertex in a path these function typically attempt to importance sample according to the product of $L_\mathrm{e}(\mathbf{x},\boldsymbol{\omega})$ or $W_\mathrm{e}(\mathbf{x},\boldsymbol{\omega})$ and the cosine term depending on if it is a subpath starting at the light source or at the sensor. For vertices in $\partial\mathcal{V}$, the best idea is usually to importance sample according to the product of the BSDF and the cosine term. Similarly, for vertices in $\mathcal{V}_0$ it is customary to importance sample according to the phase function.

Once a direction has been sampled, a distance along this direction must be sampled. The most common approach is to sample according to the transmittance term, since it is too complicated to account for inscattered and emitted radiance/importance along the ray (see e.g. Lafortune and Willems [1996] or Pauly [1999]). In the case of homogeneous media, a suitable density is given by

$$p'(s) = \sigma_\mathrm{t}\,e^{-\sigma_\mathrm{t}s},$$

which is simply a normalized version of the transmittance function. It is possible to sample from this density using a simple analytical formula, which can be derived using the inversion method,

$$s = -\frac{\ln(1-\xi)}{\sigma_\mathrm{t}}, \tag{4.18}$$

where ξ is a uniform random number in $[0;1)$. Let $\mathbf{x}_\mathrm{s} = \mathbf{x}_{\partial\mathcal{V}}(\mathbf{x},\boldsymbol{\omega})$ be the first point seen along the direction $\boldsymbol{\omega}$ at a distance d from $\mathbf{x}$ on the boundary. Then we will adapt the convention that if $s \geq d$ the next vertex will be chosen to be $\mathbf{x}_\mathrm{s}$, otherwise the next vertex will be $\mathbf{x}+s\boldsymbol{\omega}$. This means that the (discrete) probability of choosing the next vertex in $\mathcal{V}_0$ is

$$P_{\mathcal{V}_0} = \int_0^d \sigma_\mathrm{t}\,e^{-\sigma_\mathrm{t}s}\,\mathrm{d}s = 1-e^{-\sigma_\mathrm{t}d},$$

whereas the probability of choosing $\mathbf{x}_\mathrm{s}$ as the next vertex is

$$P_{\partial\mathcal{V}} = 1-P_{\mathcal{V}_0} = e^{-\sigma_\mathrm{t}d}.$$

Combining these probabilities with $p'(s)$ normalized to the interval $[0;d)$ leads to the final probability for propagation in homogeneous media,

$$p(s) = \begin{cases} P_{\partial\mathcal{V}} & \text{if } s \geq d, \\ \sigma_\mathrm{t}\,e^{-\sigma_\mathrm{t}s} & \text{otherwise.} \end{cases}$$

For the case of heterogeneous media, no such simple procedure exists. In this case, if the inversion method is applied, the result is the implicit equation,

$$\int_0^s \sigma_\mathrm{t}(\mathbf{x} + t\boldsymbol{\omega})\ \mathrm{d}t = -\ln(1-\xi). \tag{4.19}$$

The classical way of solving this equation is to use *ray marching* [Perlin and Hoffert, 1989]. Ray marching works by stepping along the ray in fixed increments while accumulating σ_t. This is continued until the threshold $-\ln(1-\xi)$ is reached or until $\mathbf{x}_\mathrm{s}$ is reached, whichever comes first.

As discussed by Raab, Seibert, and Keller [2007], this approach is biased. A solution is also presented in this paper, which depends on being able to bound $\sigma_\mathrm{t}(\mathbf{x})$ along the ray segment. First, let $\sigma_\mathrm{t} = \sup_{s\in[0,d)} \sigma_\mathrm{t}(\mathbf{x} + s\boldsymbol{\omega})$ be the maximum extinction coefficient along the ray. Then, the proposed algorithm is as follows. Let $t_1, t_2, \ldots$ be independent random distances sampled using Equation 4.18. Furthermore, let $\xi_1, \xi_2, \ldots$ be uniformly distributed random numbers in $[0; 1]$. Then, the first distance $s_i = \sum_{j=1}^{i} t_j$ which satisfies $\xi_i \leq \sigma_\mathrm{t}(\mathbf{x}+s_i\boldsymbol{\omega})/\sigma_\mathrm{t}$ is distributed according to Equation 4.19.

It is also necessary to have a way of determining when to terminate a random walk. Recall that in order to solve Equation 4.17 in an unbiased way there must be a nonzero probability of sampling any path that transmits light to the sensor, regardless of the length of the path. Consequently, simply restricting paths to a maximum length will lead to biased results. Instead, *Russian roulette*, introduced to the graphics community by Arvo and Kirk [1990], can be used. Russian roulette is an unbiased technique and works by terminating paths randomly with a predefined probability. To compensate, if a path survives it is given an additional weight of $1/q$, where q is the probability of survival. To see that this is unbiased, note that it does not change the expected value of the integral,

$$(1-q) \times 0 + q \times \frac{I}{q} = I.$$

This strategy works well to reduce the number of long paths. It can however increase the variance of the estimator, though this is usually offset by the reduction in the number long paths that needs to be considered.

If the probabilities for scattering, propagation, and Russian roulette are combined, we get the following expression,

$$p(\mathbf{x}_{i+1}|\mathbf{x}_i) = p'(\mathbf{x}_{i+1}|\mathbf{x}_i)\ p(s)\ q_{i+1},$$

for the probability of sampling $\mathbf{x}_{i+1}$ as the next vertex given that $\mathbf{x}_i$ is the current vertex. Due to the survival probability, q_{i+1}, this PDF is subcritical, i.e. it integrates to less than one.

Finally, once we know the probability of sampling the first vertex in a subpath and the probability of sampling the following vertices, the probability of the entire subpath, $\bar{\mathbf{x}} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_k$, can be computed as

$$p(\bar{\mathbf{x}}) = p(\mathbf{x}_0)\, p(\mathbf{x}_1|\mathbf{x}_0)\, p(\mathbf{x}_2|\mathbf{x}_1)\ \ldots\ p(\mathbf{x}_k|\mathbf{x}_{k-1}).$$

Connecting Subpaths

The last rule of local path sampling stated that new paths could be formed by connecting existing subpaths. Since subpaths are usually generated independently, computing the probabilities is fairly straightforward. If $\bar{\mathbf{x}}_1 = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_k$ and $\bar{\mathbf{x}}_2 = \mathbf{x}_{k+1}\mathbf{x}_{k+2}\mathbf{x}_{k+3}\ldots\mathbf{x}_{k+l}$ are two independently generated subpaths, then the probability of the path formed by connecting the two subpaths is simply

$$p(\bar{\mathbf{x}}_1\bar{\mathbf{x}}_2) = p(\bar{\mathbf{x}}_1)\, p(\bar{\mathbf{x}}_2).$$

Implications of using Local Path Sampling

Local path sampling forms the basis of numerous global illumination algorithms, including all the known unbiased algorithms. One of the requirements of an unbiased estimator is that there must be a positive probability of generating samples wherever the integrand is nonzero, which in terms of light transport is translated into that every path that transports light from the light source to the sensor must be sampled with positive probability. In order to investigate the unbiasedness of such algorithms, we therefore need a way to reason about the limitations of local path sampling with respect to what paths that can be sampled using this approach.

A useful tool for investigating these limitations is the regular expression framework developed Heckbert [1990]. Though originally developed to classify paths for regular surface light transport, the framework is easily extended to also handle participating media. The idea is to classify each vertex in a path according to whether the scattering event at the vertex is specular (S) or diffuse (D). Here we will adopt the convention that specular means that the scattering kernel (BSDF or phase function) that resulted in the scattering event includes a Dirac delta function. All remaining vertices are classified as diffuse.

The original framework of Heckbert [1990] only describes scattering events at the interior vertices of a path connecting a sensor and a light source and not at the two endpoints. To also describe the scattering event at the first and last vertex, i.e. at the light source and sensor, the full path regular expressions described in

Veach [1997] can be used. Scattering at these vertices can be described using two letter combinations. For vertices on light sources, the first letter will be S (specular) if $L_e(\mathbf{x}, \boldsymbol{\omega})$ has a Delta function with respect to the spatial component $\mathbf{x}$ and otherwise D (diffuse). Similarly, the second letter will be S if $L_e(\mathbf{x}, \boldsymbol{\omega})$ has a Delta function with respect to the directional component $\boldsymbol{\omega}$ and diffuse otherwise. Using this convention, a point light source would be described by LSD and an area light source by LDD. Cameras, or sensors, can be described in a similar way using $W_e(\mathbf{x}, \boldsymbol{\omega})$, so that DSE is a pinhole camera, and DDE is a camera with finite aperture (see Veach [1997, pg. 230] for further details).

Using these conventions, *any* path can be described by a regular expression of the form[1]

$$L\,(S|D)^*\,E,$$

where parentheses signify grouping, '$*$' implies zero or more repetitions of the string, and '|' means alternation. However, any path generated using local path sampling is necessarily of the form

$$L\,(S|D)^*\,D\,D\,(S|D)^*\,E, \tag{4.20}$$

which is clearly a subset of all possible paths (see Veach [1997, pg. 237] for a proof of this claim). The reason that two consecutive diffuse vertices are required is that if at least one of the vertices was specular then the scattering function of this vertex would only be nonzero for single directions, technically sets of measure zero, and finding these directions randomly is impossible. A consequence of this is that any algorithm based on local path sampling will be biased if the scene contains paths with $f(\bar{\mathbf{x}}) > 0$ that cannot be described by Equation 4.20. Since local path sampling cannot find these paths, the effect of these paths will be missing and the resulting image will be too dark.

A simple way to avoid this problem is to disallow specular vertices altogether. This would mean that perfect specular reflection and refraction, pinhole cameras, etc., would have to be approximated. E.g., perfect specular reflection could be approximated using the BRDF by Cook and Torrance [1982] with a very low roughness. A more practical alternative is too only disallow that the first and last vertices in a path are specular. This means that the renderer would only support area light sources ($L\,DD$) and/or cameras with finite aperture ($DD\,E$). Several commercial renderers that claim unbiasedness seem to follow this route.

[1] With these regular expressions it is understood that their length must always be at least four, since $L\,X\,X\,X\,X\,E$ corresponds to a path of length one, which is the shortest possible path.

4.7 Existing Algorithms

Local path sampling forms the basis of most newer global illumination algorithms. This includes algorithms based on photon mapping (e.g. Jensen [2001] and more recently Hachisuka, Ogaki, and Jensen [2008]) and algorithms based on virtual point light sources, such as Keller [1997] and Walter, Fernandez, Arbree, Bala, Donikian, and Greenberg [2005], and countless other algorithms.

However, as discussed in Chapter 1, we will restrict our attention to the subset of global illumination algorithms that are unbiased.

4.7.1 Path Tracing

Path tracing is generally considered the earliest unbiased global illumination algorithm. The algorithm was introduced in the seminal paper by Kajiya [1986], which is arguably the most cited paper in computer graphics. This paper also introduced the rendering equation, which is the equation that describes the distribution of radiance in the absence of participating media (it is essentially the boundary conditions for the equation of transfer introduced earlier). The introduction of the rendering equation was important, since it provided a way of identifying the limitations of earlier rendering algorithms, such as classical ray tracing [Whitted, 1980], distribution ray tracing [Cook et al., 1984], and radiosity algorithms [Goral et al., 1984].

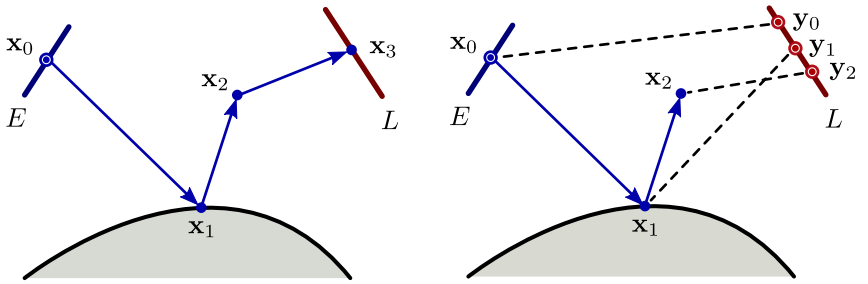
Path tracing constructs paths by sampling the first vertex on the sensor. The remaining vertices are sampled according to the local path sampling procedure described previously. This is illustrated in Figure 4.2. Since the branching factor is one at each vertex, this results in a path, rather than a tree as in classical ray tracing, which is the reason for the name of the algorithm.

Two variations of the path tracing algorithm exists. The simplest is pure path tracing, shown in Figure 4.2a. This variation samples random paths of the form, $\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$, which results in the estimator

$$I \approx \frac{f(\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k)}{p(\mathbf{x}_0)\, p(\mathbf{x}_1|\mathbf{x}_0)\, p(\mathbf{x}_2|\mathbf{x}_1) \, \dots \, p(\mathbf{x}_k|\mathbf{x}_{k-1})}.$$

Due to importance sampling, many of the terms in the measurement contribution function will get canceled by corresponding terms in the denominator. The notable exception is that the $L_\mathrm{e}(\mathbf{x}_k \rightarrow \mathbf{x}_{k-1})$ term is not matched by any corresponding term in the denominator, since paths are generated completely independently of the light sources in the scene. A consequence of this is that

Figure 4.2: The path tracing algorithm samples paths starting at the sensor (blue line marked E in the figure). We will adopt the conventation that blue arrows/circles mean that the subpath was sampled starting from the sensor and red arrows/circles indicate that it was started from the light source. Two concentric circles indicate that this vertex is the initial vertex in the path. Pure path tracing (left) generates paths $\mathbf{x}_0\mathbf{x}_1\ldots\mathbf{x}_k$, which only contribute if one of the vertices fall on a light source (red line marked L in the figure). Path tracing with next event estimation (right) explicitly connects non-specular vertices in the eye path to random points, $\mathbf{y}_n$, on the light source generating path of the form $\mathbf{x}_0\mathbf{x}_1\ldots\mathbf{x}_k\mathbf{y}_n$. Visibility between the connecting vertices is checked by tracing a shadow feeler, which is illustrated using dashed lines. Using next event estimation is typically more efficient, but risks generating a weak singularity if a path randomly gets very near a light source.



(a) Pure path tracing. **(b)** Path tracing with next event estimation.

this estimator can have very high variance even for simple scenes. The reason is that for typical scenes $L_e(\mathbf{x}_k \to \mathbf{x}_{k-1})$ is zero for most of $\mathcal{V}$. This means that there will only be a contribution if a path randomly hits a light source, which is unlikely if the light sources are small.

One cause of this inefficiency is that pure path tracing does not use the full framework provided by local path sampling. As a result, pure path tracing can only sample paths of type

$$L\,DD\,(S|D)^*\,E,$$

i.e., only area light sources are supported and the effects of point light sources and directional light sources will simply be missing. The efficiency of path tracing can be improved by using *direct lighting* techniques (also known as *next event estimation*, see Lafortune [1995] or Dutre [1996]). The resulting algorithm, illustrated in Figure 4.2b, also samples paths starting at the sensor, but at each non-specular vertex attempts to form a complete path by sampling a

random point on a light source and checking visibility. The resulting algorithm can sample paths of type

$$L\,(S|D)\,DD\,(S|D)^*\,E,$$

which means that point light sources can now be handled. However, some effects will still be missing, such as caustics from point light sources. Using direct lighting leads to paths of the form $\mathbf{x}_0\mathbf{x}_1\ldots\mathbf{x}_k\mathbf{y}_n$ and estimators

$$I \approx \frac{f(\mathbf{x}_0\mathbf{x}_1\ldots\mathbf{x}_k\mathbf{y}_n)}{p(\mathbf{x}_0)\,p(\mathbf{x}_1|\mathbf{x}_0)\,p(\mathbf{x}_2|\mathbf{x}_1)\;\ldots\;p(\mathbf{x}_k|\mathbf{x}_{k-1})\,p(\mathbf{y}_n)}.$$

These estimators are often much more efficient, since now the $L_\mathrm{e}(\mathbf{x}_k \to \mathbf{x}_{k-1})$ term is also matched in the denominator, assuming $\mathbf{y}_n$ is sampled according to emitted light. Much more sophisticated methods of doing direct lighting are also possible; see e.g. Shirley [1991].

Though the direct lighting optimization can greatly improve the efficiency, it can in some cases compromise the stability of the algorithm compared to pure path tracing. The reason is that the inverse square distance hidden in the geometry term in the measurement contribution function, $G(\mathbf{x}_k \leftrightarrow \mathbf{y}_n)$, is not matched in the denominator when next event estimation is used. As discussed by Kollig and Keller [2004], this leads to a weak singularity and potentially infinite variance. The authors present a solution based on bounding the geometry term and compensating for the resulting bias by sampling a random direction according to the BSDF or phase function. However, this can only be used to avoid weak singularities from area light sources; point light sources can still cause problems.

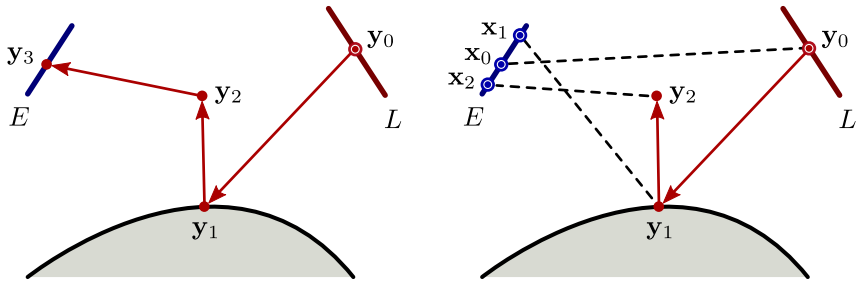
The adjoint photon tracing algorithm presented in Morley, Boulos, Johnson, Edwards, Shirley, Ashikhmin, and Premože [2006] is a variation of path tracing designed to overcome some of these problems. The algorithm is based on pure path tracing (i.e. no direct lighting), but uses a mixture density based on the BSDF or phase function, and the light sources to choose new directions at each vertex. The final probabilities are computed using multiple importance sampling. In addition, adjoint photons only carry one wavelength. This avoids the infinite variance that could result from wavelength dependent scattering, such as Rayleigh scattering.

4.7.2 Light Tracing

The "adjoint" of the path tracing algorithm is called light tracing [Dutré et al., 1993]. As shown in Figure 4.3, this algorithm samples paths using random walks starting at the light sources. Like path tracing, light tracing comes in two variations depending on whether or not next event estimation is used.

Pure light tracing, illustrated in Figure 4.3a, is a direct simulation of how particles are scattered in an environment and randomly end up on the sensor. In some sense, this is a more natural approach than path tracing, since it more accurately mirrors the underlying physical process of transporting light energy from the light source to the sensor.

Figure 4.3: Light tracing can be seen as the adjoint of the path tracing algorithm, since it samples paths starting at the light sources. Pure light tracing (left) generates paths $\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_k$, which only contribute if a ray randomly intersects the sensor. Light tracing with next event estimation (right) explicitly connects non-specular vertices to random points, $\mathbf{x}_n$, on the sensor generating path of the form $\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_k\mathbf{x}_n$.



(a) Pure light tracing. **(b)** Light tracing with next event estimation.

Unfortunately, this approach can be very inefficient, since it relies on randomly hitting the set of rays where $W_e(\mathbf{y}_{k-1} \to \mathbf{y}_k) > 0$, which may be arbitrarily small. For instance, a typical camera has an aperture with a radius of only a few mm, which makes it a very small target in typical scenes. In fact, since the path must intersect the sensor to form a complete path, only cameras with finite aperture can be simulated this way. I.e., only paths of type

$$L\,(S|D)^*\,DD\,E$$

can be sampled. Using pure light tracing results in an estimator of the form

$$I \approx \frac{f(\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_k)}{p(\mathbf{y}_0)\,p(\mathbf{y}_1|\mathbf{y}_0)\,p(\mathbf{y}_2|\mathbf{y}_1)\,\dots\,p(\mathbf{y}_k|\mathbf{y}_{k-1})}.$$

As shown in Figure 4.3b, adding next event estimation is analogues to the path tracing case. For each non-specular vertex, a random point is selected on the aperture and visibility is checked. This leads to paths, $\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_k\mathbf{x}_n$, of type

$$L\,(S|D)^*\,DD\,(S|D)\,E,$$

which means that pinhole cameras can now be properly accounted for. The estimator for light tracing with next event estimation is

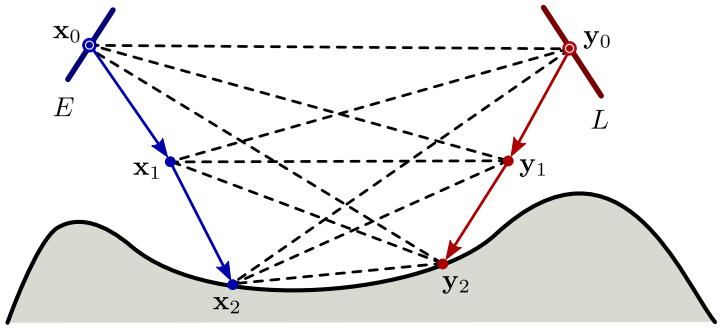
$$I \approx \frac{f(\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_k\mathbf{x}_n)}{p(\mathbf{y}_0)\, p(\mathbf{y}_1|\mathbf{y}_0)\, p(\mathbf{y}_2|\mathbf{y}_1)\, \dots\, p(\mathbf{y}_k|\mathbf{y}_{k-1})\, p(\mathbf{x}_n)}.$$

This estimator suffers from the same problem with weak singularities that the path tracing estimator does. One fundamental difference from path tracing is that where a given path family generated with path tracing only contributes to pixel(s) near the start point of the path, the paths generated with light tracing can influence many different pixels across the image plane.

4.7.3 Bidirectional Path Tracing

For most scenes, path tracing is a more efficient algorithm than light tracing. However, certain paths are very difficult to find starting from the sensor, and consequently light tracing is better at finding these paths. Bidirectional path tracing is an algorithm that attempts to preserve the strengths of both these algorithms by sampling paths from both directions. Bidirectional path tracing was developed independently by Lafortune and Willems [1993] and Veach and Guibas [1994] and extended to participating media by Lafortune and Willems [1996].

Figure 4.4: Bidirectional path tracing samples paths by first performing two random walks; the first path, $\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$, starting from the sensor and the second path, $\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_l$, starting from the light sources. Complete paths are then formed by connecting a vertex from the eye path to a vertex from the light path.



The general idea behind bidirectional path tracing is illustrated in Figure 4.4. In this algorithm, for each sample two random walks are performed; the first

starting from the sensor and the second from the light source. Complete paths are formed by connecting a vertex from the eye path to a vertex on the light path. This requires computing visibility and transmittance between the two vertices, similar to the process of next event estimation. Bidirectional path tracing can sample paths of the form

$$L\,(S|D)^*\,D\,D\,(S|D)^*\,E,$$

i.e. any path that can be sampled with local path sampling. This makes it a more general algorithm than path tracing or light tracing. However, since bidirectional path tracing is based on local path sampling, there are still some paths it cannot sample.

Consider a sensor subpath, $\bar{\mathbf{x}} = \mathbf{x}_0\mathbf{x}_1 \ldots \mathbf{x}_{t-1}$, with t vertices and a light source subpath, $\bar{\mathbf{y}} = \mathbf{y}_0\mathbf{y}_1 \ldots \mathbf{y}_{s-1}$, with s vertices. If the endpoints of these paths are connected the path, $\bar{\mathbf{z}} = \mathbf{y}_0\mathbf{y}_1 \ldots \mathbf{y}_{s-1}\mathbf{x}_{t-1}\mathbf{x}_{t-2} \ldots \mathbf{x}_0$, of length $k = s + t - 1$ is formed. An estimator for this path is given by

$$I \approx w_{s,t}(\bar{\mathbf{z}})\,\frac{f(\bar{\mathbf{z}})}{p(\bar{\mathbf{x}})\,p(\bar{\mathbf{y}})},$$

where $p(\bar{\mathbf{x}}) = p(\mathbf{x}_0)\,p(\mathbf{x}_1|\mathbf{x}_0)\,p(\mathbf{x}_2|\mathbf{x}_1)\,\ldots\,p(\mathbf{x}_{t-1}|\mathbf{x}_{t-2})$ is probability of generating path $\bar{\mathbf{x}}$ and analogously for $p(\bar{\mathbf{y}})$.

The function $w_{s,t}(\bar{\mathbf{z}})$ is the weight of the path. Recall that we are trying to solve Equation 4.16 by generating samples from each integral. However, as shown in Figure 4.4, a single sample generated using bidirectional path tracing results in a family of paths with $k + 2$ paths of length k, if Russian roulette and specular vertices are ignored. This means that the contribution from each path needs to be weighted according to how long the path is.

Rather than using constant weights, much better results can be achieved if the weight is allowed to depend on how the path was generated. The key insight by Veach [1997] is that a path of length k can be generated using $k + 2$ methods, corresponding to how many vertices there are in the sensor subpath versus how many in the light subpath. These $k+2$ methods generate the same path but with different probabilities. This makes bidirectional path tracing an ideal candidate for applying the technique of multiple importance sampling (Veach and Guibas [1995], see Section 3.7.1 for a general description of this technique). For instance, if the balance heuristic is used, the weighting function is given by

$$w_{s,t}(\bar{\mathbf{z}}) = \frac{p_{s,t}(\bar{\mathbf{z}})}{\sum_{i=0}^{s+t} p_{i,s+t-i}(\bar{\mathbf{z}})}, \tag{4.21}$$

where $p_{s,t}(\bar{\mathbf{z}})$ is the probability of generating the path $\bar{\mathbf{z}}$ using s vertices in the light source subpath and t vertices in the sensor subpath. In Equation 4.21, the

numerator is the probability with which the path was actually generated, i.e. $p(\bar{\mathbf{x}})\,p(\bar{\mathbf{y}})$. In the denominator, the summation is over all the possible ways the path could have been generated, which can generally happen in $k+2$ ways, but can be fewer if the path has specular vertices.

It is also possible to use other heuristics. Veach [1997, pg. 306] suggests using the power heuristic with $\beta = 2$, and shows that this heuristic is often the best choice. Another alternative is to use the maximum heuristic, which has the advantage that the number of shadow rays can be reduced, since only the most likely method of generating a path has to be considered. The problem of reducing the number of shadow rays is also considered in Lafortune and Willems [1995a] and in Veach [1997, pg. 317]. Kollig and Keller [2004] presents a weighting function based on bounding the geometry term, and shows that this method can outperform weighting functions based on multiple importance sampling in some cases.

4.7.4 Markov Chain Monte Carlo Ray Tracing

Markov Chain Monte Carlo has also been applied to solving the light transport problem. The Metropolis-Hastings algorithm, presented in Section 3.8.2, was used in the Metropolis light transport algorithm by Veach and Guibas [1997] (with many more details in the thesis by Veach [1997, chp. 11]). This algorithm was later extended to participating media by Pauly [1999] and Pauly et al. [2000].

The idea is to use the Metropolis-Hastings algorithm to directly sample paths in path space using the measurement contribution function, $f(\bar{\mathbf{x}})$, as the desired stationary distribution of the chain,

$$\pi^*(\bar{\mathbf{x}}) = \frac{f(\bar{\mathbf{x}})}{b} \tag{4.22}$$

where $b = \int_\Omega f(\bar{\mathbf{x}})\,\mathrm{d}\mu(\bar{\mathbf{x}})$ is a normalization constant. This target density is chosen, since it is a good choice from an importance sampling point of view.

This formulation leads to a Markov chain, where each state corresponds to a path connecting the light source and sensor. One complication with this approach is that we are trying to compute an image with many pixels and each pixel has its own flux responsivity function, W_{e}^j, and thus its own measurement contribution function. A simple way to solve this problem is to define a global flux responsivity function by combining the n flux responsivity functions in some way, such as by taking their sum

$$W_{\mathrm{e}}(\mathbf{x}, \boldsymbol{\omega}) = W_{\mathrm{e}}^1(\mathbf{x}, \boldsymbol{\omega}) + W_{\mathrm{e}}^2(\mathbf{x}, \boldsymbol{\omega}) + \ldots + W_{\mathrm{e}}^n(\mathbf{x}, \boldsymbol{\omega}).$$

The global flux responsivity function can then be used in the measurement contribution function used in Equation 4.22 to ensure that all measurements get sampled.

The result is that when a path has been generated during rendering, the pixels that this path contributes to have to be identified and their accumulators incremented. What this essentially means is that an image is formed by making a histogram of how the path endpoint moves in the image plane. I.e., bright pixels are more intense because more paths have been generated that pass through them. This is contrary to how path tracing and bidirectional path tracing work. In those algorithms bright pixels are caused by randomly finding high energy paths using a typically fixed number of samples per pixel.

Initializing the Markov chain can be tricky. This is the problem of startup bias discussed in Section 3.8.2. The problem is that it is generally not possible to sample the initial state according to Equation 4.22, since this is the reason we are resorting to the Metropolis-Hastings algorithm in the first place. Instead, a random initial state is chosen and the chain is allowed to burn in by ignoring the k first states, which effectively causes the chain to forget its starting point. The problem with this approach is that determining k is difficult. It is also wasteful, since the k first samples are ignored. The Metropolis light transport algorithm solves the initialization problem using a two step procedure that results in a population of seed paths distributed according to Equation 4.22. This is done by first sampling a large number of paths using bidirectional path tracing. A smaller number of paths are then resampled from the large population with probability $f(\bar{\mathbf{x}})/p(\bar{\mathbf{x}})$, where $p(\bar{\mathbf{x}})$ is probability with respect to path space. This results in a number of seed paths with the correct distribution that can be used as the initial states of the chains. See Szirmay-Kalos, Dornbach, and Purgathofer [1999] for further discussion of the problems of startup bias.

Once the path corresponding to the initial state has been found, Algorithm 3.2 is used to evolve the chain and generate the next states. A tentative sample is first generated by mutating the current path according to a set of predefined mutation strategies. The tentative sample is then probabilistically accepted as the next state, or rejected, which means that the chain stays in the current state. Veach and Guibas [1997] suggest a number of strategies for mutating the current path. Bidirectional mutations modify a path by replacing a random subpath by a new subpath. This mutation type causes large changes to the current path, and thus helps improve the mixing rate, while also ensuring the ergodicity of the chain. Scattering perturbations are mutation types that favor small changes to the path by altering the outgoing direction at a vertex. These mutations are effective at handling certain difficult lighting situations, most notably caustics. Finally, Pauly et al. [2000] suggest a propagation perturbation designed to handle participating media. Like scattering perturbations, propa-

gation perturbations favor small changes and thus explore locally around the current path. As discussed in Section 3.8.2, this ability to preserve the sampling context is the background for the success of the Metropolis-Hastings algorithm.

More recently, Kelemen, Szirmay-Kalos, Antal, and Csonka [2002] introduced a new formulation of Metropolis light transport based on mutating the random numbers used to generate a path, rather than mutating the vertices of the path directly. Raab et al. [2007] discuss how to extend this variation to participating media. The idea is that the random numbers used to generate a given path can be seen as a point in the infinite dimensional unit cube, which they call the primary sample space. By perturbing the current state in the primary sample space they achieve an effect similar to the original Metropolis light transport algorithm.

The primary advantage of this formulation is its simplicity. Rather than working directly with the geometry of paths, which can be tricky, the new algorithm simply modifies the random numbers and regenerates the path, which tends to lead to a simpler algorithm. Another advantage of working in the primary sample space is the algorithm is not tied to a particular way of sampling paths. This means that any of the previously covered algorithms, such a path tracing and light tracing, can be used as the underlying algorithm to map the random numbers to path space. In particular, if bidirectional path tracing is used to map from the primary sample space to path space, then each sample corresponds to a family of paths, such as the paths in Figure 4.4. This is quite different from the original formulation of the Metropolis light transport algorithm, where a sample corresponds to a single path connecting the light and sensor. With this new formulation the target density (Equation 4.22) needs to be modified, so that it is based on the flow of light energy along all the paths in the family.

Kelemen et al. suggest two strategies for mutating the samples in the primary sample space. Large mutations generate new samples independently of the current sample by sampling a new random position in the primary sample space. This is similar to ordinary Monte Carlo, and ensures the ergodicity of the chain. Small mutations perturb the current location according to an exponential density and help explore locally around the current sample. These are in fact the independence kernel and random walk kernel discussed in Section 3.8.2. In a sense, this means that the algorithm can be described as a combination of regular Monte Carlo and Markov Chain Monte Carlo.

It is unclear which formulation of Metropolis light transport leads to the most efficient estimators, since no in-depth study of the relative merits of the two formulations has been performed. The variance of the original algorithm was studied by Ashikhmin, Shirley, and Smits [2001], but this paper predates the new formulation. Kelemen et al. [2002] provide one numerical comparison of the

efficiency of the two methods. As mentioned in that paper, their formulation has the advantage that perturbations take place before importance sampling. As discussed in Section 3.8.2, this means that the size of the mutations automatically adapt to the shape of the integrand, which reduces correlation between samples, while increasing the acceptance ratio. However, since importance sampling is not carried out after the full integrand of Equation 4.14 or 4.15 (incident radiance/importance is ignored) it is unclear how large the benefit of this really is. Also, as mentioned by Veach [1997, pg. 354], their algorithm is not particularly sensitive to the size of mutations in the first place.

Another difference between the algorithms is the cost of generating a sample. Generally, generating new samples with the original algorithm is much cheaper, since it often only requires tracing a few additional rays, whereas generating a sample with the new formulation requires tracing two random walks and connecting the vertices. This means that the new formulation risks oversampling easy to find paths while trying to sample a few hard paths, since the easy paths are automatically generated by the bidirectional path tracing algorithm.

Other algorithms exist that use path mutations. The Energy Redistribution Path Tracing algorithm by Cline, Talbot, and Egbert [2005] uses a combination of regular Monte Carlo and Metropolis-Hastings mutations. The authors claim that this leads to a simpler algorithm than the original Metropolis light transport algorithm, while still being competitive performance wise.

4.8 Discussion

The framework of local path sampling provides a simple recipe for constructing paths based on random walks. As discussed above, this leads to a number of different estimators with different properties. One of these properties is the efficiency of the estimator. As discussed in Section 3.3, the efficiency of an estimator tells us something about the variance of the resulting samples and how fast they can be generated, i.e. how fast the estimator can solve a given problem. As with any Monte Carlo estimator, the variance is caused by the inability to importance sample exactly according to the integrand, which in the present case is the measurement contribution function, $f(\bar{\mathbf{x}})$. In the remainder of this chapter we will discuss why algorithms based on local path sampling have difficulty sampling paths distributed according to this function.

The first question that needs to be answered is if it is even desirable to sample according to $f(\bar{\mathbf{x}})$ across the image plane. Recall that we are usually computing multiple measurements simultaneously, each measurement the result of a sensor

response W_e^j for $j = 1, 2 \ldots n$, which is nonzero only for some typically small subset of the image plane. If paths are sampled according to $f(\bar{\mathbf{x}})$ (based on the global flux responsivity function, W_e) then bright pixels will automatically receive more samples than dimmer pixels, and there will be a large disparity between how many samples each measurement receives. A consequence of this is that the measurements corresponding to bright pixels will get estimated with a relatively higher accuracy than measurements corresponding to dimmer pixels.

The result of this partial undersampling will be perceived by an observer as noise in the dark regions of an image. The problem is that sampling according to $f(\bar{\mathbf{x}})$ will tend to reduce absolute error. However, the human visual system is more sensitive to ratios than to absolute differences. Therefore, since images are usually intended for human viewers, it is better to sample in the image plane so as to reduce relative error or a more advanced perceptual metric. This is easy to do for an algorithm such as path tracing, since the sample positions are chosen explicitly in the image plane. This makes it relatively simple to control how many samples each measurement receives, and also makes it possible to assign more samples to measurements with high variance. However, for an algorithm such as Metropolis light transport, controlling the sample density in the image plane is more difficult. A partial solution is suggested by Veach [1997, pg. 357], which is based on computing a low resolution luminance image in a preprocess. This image is then used to modify the target density in a way that forces the sampling density to be approximately uniform across the image plane. The downside to this approach is that it can be time consuming to compute the luminance image and if the resolution is insufficient artifacts in the form of an uneven amount of noise across the image plane can appear.

4.8.1 Drawbacks of Local Path Sampling

The algorithms covered in the previous sections, such as path and light tracing, follow importance, radiance, or both, as it scatters through the scene. The random walks are created using local path sampling by first sampling a random starting point on either the sensor or light source. Additional vertices are found by first sampling a direction using the scattering kernel at the vertex and then sampling a distance along that direction based on the properties of the medium. In the following we will discuss the assumptions of local path sampling as it is typically used and the inefficiencies these assumptions cause.

The first problem that needs to be considered is how the initial vertex in a path is sampled. It turns out that deciding where to sample the first vertices on sensor paths is often easier than finding good starting points for the light paths. The reason is that there is usually only one camera, which has a nonzero flux

responsivity function for only a very small subset of $\mathcal{V}$ (or even just a single point for a pinhole camera). However, there may be many light sources in a scene, and information on which of those light sources are important to the camera is generally not available. Ideally we would like to sample starting points on light sources according to the product of incident importance and emitted radiance across the light sources, since this is essentially $f(\bar{\mathbf{x}})$. However, this is generally not possible without some sort of global context, such as knowledge of W_{i} or L_{i}.

Additional vertices are added to a path by first sampling a direction according to a suitable density, which depends on the type of the vertex. This is the scattering step of local path sampling. For the initial vertex on a light path directions are sampled according to emitted radiance or cosine weighted emitted radiance, depending on whether it is a volume or surface light source. For the initial vertex on a sensor path the direction is sampled according to the flux responsivity function defined by the camera model and for the remaining vertices the direction is sampled according the scattering kernel. Ideally, for a vertex on a light path it would be better to sample according to the product of this function and incident importance, since this would closer mimic all of the terms of $f(\bar{\mathbf{x}})$. Similarly, for vertices on the sensor path, sampling according to the product of this function and incident radiance would closer match $f(\bar{\mathbf{x}})$ (with the possible exception of the first vertex, as discussed above). However, again this is not possible without knowledge of W_{i} or L_{i}.

The second part of sampling additional vertices with local path sampling is the propagation step. As discussed in Section 4.6.1, the new vertex is sampled on a random point in the given direction or at the boundary, where the exact procedure for doing this depends on whether the medium is homogeneous or heterogeneous. In either case, for sensor paths (light paths) the sampling procedure is based on the assumption that inscattered radiance (importance) along the ray is constant and the same as the inscattered radiance (importance) at the boundary. The sampling procedure also ignores emitted radiance (importance), i.e. it assumes that the emitted radiance (importance) along the ray and the emitted radiance (importance) at the boundary is zero. As a result, whenever these assumptions are not true, the probability density is a poor match for the integrand and the result is excessive noise. Unfortunately, like above, solving these problems requires knowledge of W_{i} or L_{i}.

With local path sampling, random walks are probabilistically terminated using Russian roulette. Using this technique can be advantageous, since it limits the number of long paths that need to be considered in an unbiased way. The probability of continuing a path is often given as the albedo of the scattering kernel (the hemispherical-directional reflectance for BSDFs and $\sigma_{\mathrm{s}}(\mathbf{x})/\sigma_{\mathrm{t}}(\mathbf{x})$ for phase functions). This can lead to increased variance in cases where the incident

radiance or importance at a vertex is high, but the albedo is low. This means that ideally, the probability of terminating a path should also depend on incident radiance or importance. For instance, a light path should be terminated with high probability if it enters a region with low importance and similarly for sensor paths. However, applying this strategy also requires knowledge of W_i or L_i.

Based on the above discussion, it is clear that algorithms based on local path sampling could benefit from having some global context, such as a representation of incident radiance or importance, so that the measurement contribution function can be more efficiently importance sampled and variance reduced. It should also be clear that using more sophisticated sampling strategies that take more factors into account will generally be slower than the simple sampling strategies of regular local path sampling. As a consequence, for this to be of benefit, the increase in computation time must be more than offset by the reduction in variance, since otherwise the resulting estimator will be less efficient.

4.8.2 Drawbacks of Existing Algorithms

As discussed above, local path sampling has some limitations when it comes to sampling paths according the measurement contribution function. In this part, we will discuss how this affects the algorithms based on local path sampling that were covered earlier.

Unidirectional methods, such as path tracing and light tracing, follow either radiance or importance through the scene. This can be inefficient, since some important paths are easier to find if you start from the camera, whereas others are easier to find starting from the light source. The reason for this is the lack of global context discussed in the previous section. An example is caustics paths, which tend to be easier to find starting from the light source. Since any given scene can easily contain both types of paths, neither algorithm will perform well in these cases.

Bidirectional path tracing uses a superset of the techniques of path tracing and light tracing. This means that it can sample paths in more ways than either of those algorithms, and easily find some paths that are difficult for those algorithm to find. However, some paths are hard to find even with bidirectional path tracing or any technique based on local path sampling. The most common example of such a “hard” path is a path in a so-called near-singular configuration (see Veach [1997, pg. 240]).

Recall that in the discussion of the limitations of local path sampling in Section 4.6.1 it was found that only paths with two consecutive non-specular ver-

tices, DD, could be sampled. Paths without DD would simply be missing. A simple example of such a missing path is a point light source seen in a mirror through a pinhole camera. However, consider if the point light source is replaced by a small spherical area light source. Since this path can only be found using a random walk starting at the sensor that randomly intersects the sphere, the probability of generating the path will depend directly on the size of the sphere. If the size of the sphere is reduced, the path approaches a near-singular configuration as the probability of generating the path goes to zero. Another more advanced example of such a path is a caustic from a small light source seen indirectly through a specular surface from a pinhole camera, such as caustic on the bottom of a pool. These paths have type $LDDSDSDSE$, and can also only be generated in one way and the probability of doing so can be arbitrarily small, since it depends on the size of the light source. Near-singular configurations can also happen because of small geometric features in the scene, such as light flowing through a door slightly ajar or through a keyhole.

The algorithms based on Metropolis-Hastings are the only effective unbiased methods for handling scenes with near-singular configurations. Unfortunately, these algorithms tend to perform worse than bidirectional path tracing on the simple parts of the integrand, mostly due to lack of stratification. Bidirectional path tracing, on the other hand, tends to perform quite well in scenes without these near-singular configurations.

Chapter 5

Enlightened Local Path Sampling

In the previous chapter the classical unbiased global illumination algorithms, path tracing, light tracing, and bidirectional path tracing, were presented. These algorithms essentially provide recipes for generating samples, $\bar{\mathbf{x}}$, from path space, Ω, with known probability $p(\bar{\mathbf{x}})$. This makes it possible to solve integrals of the form

$$I = \int_{\Omega} f(\bar{\mathbf{x}}) \, \mathrm{d}\mu(\bar{\mathbf{x}}) \tag{5.1}$$

using standard Monte Carlo estimators,

$$I \approx \frac{f(\bar{\mathbf{x}})}{p(\bar{\mathbf{x}})}.$$

Common for these algorithms is that they construct paths using local path sampling (see Section 4.6.1). Paths constructed using local path sampling are typically generated independently of the relevant adjoint quantity. E.g., light paths are generated by sampling the first two vertices according to emission, L_{e}, and the remaining by sampling directions according to BSDFs or phase functions, thereby ignoring incident importance at each step (the situation for eye paths is completely analogues). Unfortunately, this is usually the only choice, since no knowledge of the incident quantity (radiance or importance) at each vertex is known, i.e. we lack some sort of global context. Because of this, the resulting probability functions, $p(\bar{\mathbf{x}})$, will in general not be proportional the measurement

contribution function, $f(\bar{\mathbf{x}})$, which would be the ideal case according to the principle of importance sampling, and consequently the resulting estimators may have high variance.

Global illumination algorithms based on Markov Chain Monte Carlo, such as the two variations of Metropolis light transport discussed in the previous chapter, often outperform regular Monte Carlo whenever $p(\bar{\mathbf{x}}) \propto f(\bar{\mathbf{x}})$ is far from being true. However, as with any Markov Chain Monte Carlo algorithm, these algorithms only follow the desired distribution in the limit. In addition, since these algorithms are still based on local path sampling, the further $p(\bar{\mathbf{x}})$ is from being proportional to $f(\bar{\mathbf{x}})$ the harder it becomes to propose good tentative samples (recall the discussion in Section 3.8 that a good sample is a sample that is very different from the current state of the chain and that is accepted with high probability, since this ensures that the state space is explored quickly). However, if $p(\bar{\mathbf{x}})$ is very different from $f(\bar{\mathbf{x}})$, most large mutations will be rejected and only the small perturbations accepted, which will cause the chain to mix slowly, resulting in slow convergence. As a consequence, even algorithms based on Markov Chain Monte Carlo will benefit from having global context, so that importance sampling can happen more effectively.

This chapter presents an algorithm that attempts to provide this missing global context. The algorithm has two parts. In the first part radiance and importance flow in the scene is analyzed and organized in suitable data structures. In the second part, the scene rendered using a variation of the Metropolis light transport algorithm modified to use this data structure.

The remainder of this chapter is organized as follows. In the first section we present an overview of the proposed algorithm. In the next sections we discuss how to provide global context and how to use this extra information to improve rendering. In the last sections we present results of using this algorithm and compare its performance to other algorithms. We conclude with a discussion of possible improvements.

5.1 Algorithm Overview

The global context required to sample according to $f(\bar{\mathbf{x}})$ is the equilibrium distribution of incident radiance, $L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega})$, and the equilibrium distribution of incident importance, $W_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega})$, in the scene, or an approximation thereof. If this information is known, it is, at least theoretically, possible to generate paths with a probability distribution based on all the terms of the measurement contribution function.

5.1.1 Local Path Sampling Using Global Context

To make this more concrete, consider generating a light path as part of doing bidirectional path tracing. With traditional local path sampling, the initial vertex, $\mathbf{x}_0 \in \partial\mathcal{V}$, is typically sampled with density

$$p(\mathbf{x}_0) \propto \int_{\mathcal{S}^2} L_\mathrm{e}(\mathbf{x}_0, \boldsymbol{\omega}) \,|\cos\theta|\, \mathrm{d}\sigma(\boldsymbol{\omega}), \tag{5.2}$$

i.e., proportional to radiant exitance. However, if the importance equilibrium distribution is known, the following density can be used instead,

$$p(\mathbf{x}_0) \propto \int_{\mathcal{S}^2} L_\mathrm{e}(\mathbf{x}_0, \boldsymbol{\omega})\, W_\mathrm{i}(\mathbf{x}_0, \boldsymbol{\omega}) \,|\cos\theta|\, \mathrm{d}\sigma(\boldsymbol{\omega}). \tag{5.3}$$

The use of importance in an inner product with an exitant quantity (L_e) makes it clear that it is the importance equilibrium distribution in incident form, rather than exitant form, that is needed (similarly, the equilibrium distribution of radiance in incident form will be needed for creating eye paths).

The difference between Equation 5.2 and Equation 5.3 is that the latter takes both the intensity of the light source and how important the emitted light is to the camera into account. This could potentially reduce variance in some scenes. For instance, consider a scene with many light sources, where only a small fraction contribute light that is visible to the camera. If the initial vertices of the light paths are sampled according to Equation 5.2, most of the resulting paths will be wasted, since they will be in regions of the scene unimportant to the camera. On the other hand, if the initial vertices are sampled according to Equation 5.3, most light paths will explore important regions and it will be more likely that full paths that connect the camera and light source can be made.

The same principle can be applied when sampling the remaining vertices in the light path. For instance, if the nth vertex is $\mathbf{x}_n \in \partial\mathcal{V}$, then the next vertex, $\mathbf{x}_{n+1}$, is chosen by first sampling an outgoing direction according to the BSDF product function,

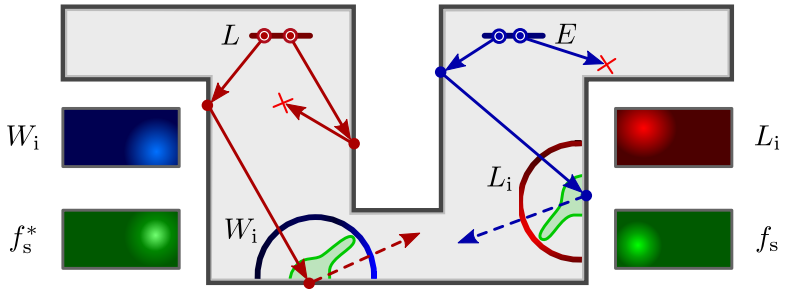
$$p(\boldsymbol{\omega}') \propto f_\mathrm{s}^*(\mathbf{x}_n, \boldsymbol{\omega}', \boldsymbol{\omega}) \,|\cos\theta|.$$

The danger of using this density is that the sampled directions may lead the path to unimportant regions of scene. As shown in Figure 5.1, if the incident importance is included in the density,

$$p(\boldsymbol{\omega}') \propto f_\mathrm{s}^*(\mathbf{x}_n, \boldsymbol{\omega}', \boldsymbol{\omega})\, W_\mathrm{i}(\mathbf{x}_n, \boldsymbol{\omega}') \,|\cos\theta|,$$

the risk of this happening is reduced, since the random walk will be more inclined to move toward the camera. This approach can also be used if $\mathbf{x}_n \in \mathcal{V}_0$, such

Figure 5.1: A simple scene where the sensor and the light source are located in two different rooms connected by a small opening. Regular local path sampling creates random walks by sampling the outgoing direction at each vertex according to the BSDF product function, phase function, or emission. This can be inefficient if the distribution of radiance and the distribution of importance are dissimilar. If incident importance/radiance (show as blue/red semicircles) is also taken into account when sampling the outgoing direction, light paths will be more likely reach areas visible to the camera, and eye paths will be more like to reach brightly lit areas. Since this approach consider more terms from the measurement contribution function, importance sampling should be improved. Incident quantities can also be used to improve Russian roulette, so that random walks that enter unimportant parts of the scene can be terminated (indicated with red crosses) with higher probability.



as if $\mathbf{x}_0$ is a volumetric light source or $\mathbf{x}_n$ a phase function. The only difference would be that the cosine terms would disappear.

Random walks are terminated using Russian roulette. The decision to terminate a path is typically based on the albedo at $\mathbf{x}_n$. If $\mathbf{x}_n \in \partial\mathcal{V}$ the albedo can be computed as the ratio $f_\mathrm{s}^*(\mathbf{x}_n, \boldsymbol{\omega}', \boldsymbol{\omega}) \,|\cos\theta| \,/\, p(\boldsymbol{\omega}')$ (assuming $p(\boldsymbol{\omega}')$ is proportional to the BSDF product function). If $\mathbf{x}_n \in \mathcal{V}_0$ the albedo is simply $\sigma_\mathrm{s}(\mathbf{x}_n)/\sigma_\mathrm{t}(\mathbf{x}_n)$. In both cases, the decision is based on an assumption of uniform incident importance and may therefore also benefit from incorporating the incident importance, so that paths that enter areas of scene that are unimportant to the sensors are terminated with higher probability (see Figure 5.1).

The remaining steps of the local path sampling procedure could potentially also benefit from having global context. In case of participating media, the propagation step is used to find the next vertex along the ray. Here, the standard approach is to sample the distance according to a 1D density which is proportional to the transmittance of the ray. This is essentially based on an assumption

of uniform incident importance. When this assumption is false, the variance of the estimator will be high. In such cases it may be advantageous to incorporate the inscattered importance along the ray and incident importance at the boundary into the density.

In the example above it was shown how light paths used in bidirectional path tracing could be sampled using knowledge of incident importance. Generating eye paths using global context is similar, but instead uses the incident radiance. The equations are the same; simply replace W_i with L_i, W_e with L_e, and the adjoint BSDF, f_s^*, with the ordinary BSDF, f_s.

Using incident radiance or importance changes the probability density functions that are used for sampling the vertices in the paths, but it does not fundamentally change the local path sampling procedure. I.e., even if other PDFs are used, the algorithms are still based on the local path sampling framework. This means that all of the algorithms based on local path sampling, including path tracing, light tracing, bidirectional path tracing, as well as the algorithms based on Markov Chain Monte Carlo, can be used with this approach.

5.1.2 Building Global Context

In order to apply the strategy outlined above, it should be possible to query incident radiance/importance at any point the scene. L_e and W_e are given explicitly as part of the scene description. However, L_i and W_i are only given implicitly in terms of L_e and W_e and the scattering properties of the scene and will therefore have to be computed in a preprocess and the results stored in a suitable data structure.

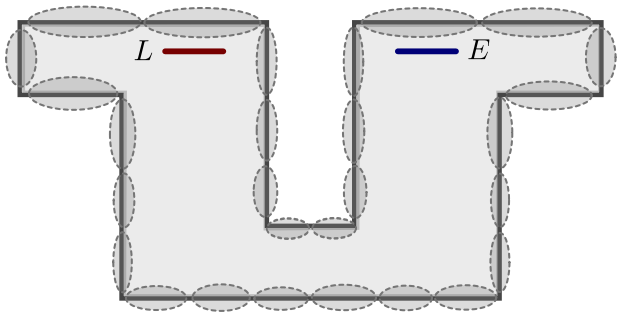
It is possible to construct an unbiased representation of L_i and W_i using a set of samples, which are essentially weighted rays. These samples are called "photons" when they represent radiance and "importons" when they represent importance, or simply particles. The classical approach for creating these particles is to perform a random walk using a procedure similar to local path sampling. Generating these particles in an intelligent way turns out to be an interesting subproblem that we will study in some detail later in this chapter. In our case, both radiance and importance is part of the required global context, so both types of particles will have to be generated. Once the particles have been created, they will need to be organized in a data structure suitable for use with the algorithm described above.

There are three main requirements for such a data structure. Firstly, the data structure should provide a faithful representation of radiance or importance,

while still being compact enough to fit in main memory. This will invariably involve a trade off between accuracy of representation and storage requirements. Secondly, querying the data structure should be fast. That is, given a point $\mathbf{x} \in \mathcal{V}$, it should be possible to quickly locate the 2D slice of the 5D function that describes incident radiance or importance at that point. Finally, the last requirement is that it should be possible to importance sample as outlined in the previous section.

Our approach is based on constructing a representation of $L_\mathrm{i}/W_\mathrm{i}$ that is piecewise constant with respect to position (see Figure 5.2). We first divide the scene into a number regions of varying size. For each region we compute the average incident radiance or importance across the region, similar to an environment map. When we create the regions, we ensure that their size depend on how much the paths that pass through those parts of the scene contribute to the image. This means that the regions in the most important parts of the scene will be smaller and consequently radiance or importance we be approximated relatively more accurately in those regions. We find the regions and estimate the relevant quantities using a form of particle tracing that samples photons and importons simultaneously, and which takes both radiance and importance into account.

Figure 5.2: We approximate incident radiance or importance by dividing the surfaces of the scene into a number of regions (shown here as dashed ellipsoids). Inside each region we store an environment map with the average incident radiance/importance across the region. Rather than using constant size regions, we allow the size of the regions in a given part of the scene to depend on how much paths that pass through that part influence the final image.



In summary, the algorithm described above is a two-part procedure consisting of a preprocessing pass and a rendering pass. In the preprocessing pass a representation of the flow of radiance and importance in the scene is created.

This extra information, or global context, is then used in the rendering pass to implement a more intelligent form of path sampling that attempts to match more terms from the measurement contribution function and thus provide more efficient importance sampling. In the following, each part of the algorithm will be described in more detail.

5.2 Preprocessing Pass

The goal of the preprocessing pass is to construct a compact representation of radiance and importance. Storing the equilibrium distribution of radiance or importance for an arbitrary scene is challenging for a number of reasons. The primary reason is that L_i and W_i are high-dimensional functions. E.g., in the form used in the previous chapters, radiance, $L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}, \lambda, t)$, is a seven dimensional function of position, direction, wavelength, and time and importance likewise.

In order to simplify the problem, and make storage requirements more practical, the dimensionality of these functions will have to be reduced. The wavelength dependence can be removed if spectral radiance or importance is replaced with a wavelength averaged quantity by integrating $L_\mathrm{i}(\lambda)$ over the visible spectrum. Optionally, a weighting function, such as the spectral luminous efficiency function, $V(\lambda)$, can be used. This has been done in all the results in this chapter.

The time dependence can also be dropped, and time averaged radiance (or importance) used instead. This is assumption is usually justified for ordinary scenes, since the shutter time of typical cameras is relatively short compared to the speed of most everyday objects. This means that except for scenes with very strong motion blur, both radiance and importance can be assumed to be nearly constant with respect to time

However, even with these assumptions, creating an exact representation of L_i or W_i in an explicit form, such as using a finite number of basis functions, is not possible in the general case, since these functions are not bandlimited. Fortunately, having an exact representation of L_i and W_i is not a requirement for the algorithm described above. Recall that L_i and W_i are only used for creating better probability density functions. This means that even if approximations of L_i or W_i are used, then as long as the resulting PDFs are valid in a Monte Carlo sense, the algorithm will remain unbiased. However, it should be expected that the poorer these approximations are the less effective importance sampling will become.

5.2.1 Adjoint Measurements

Recall that rather than sampling the initial vertex in a light path according to radiant exitance, it should be possible to achieve better results by sampling according to Equation 5.3, i.e. according to

$$p(\mathbf{x}_0) \propto \int_{\mathcal{S}^2} L_\mathrm{e}(\mathbf{x}_0, \boldsymbol{\omega})\, W_\mathrm{i}(\mathbf{x}_0, \boldsymbol{\omega})\, |\cos\theta|\, \mathrm{d}\sigma(\omega), \tag{5.4}$$

since this density takes both radiance and importance into account. While a similar approach could be used for sampling the initial vertex in the eye path, i.e. sampling according to

$$p(\mathbf{y}_0) \propto \int_{\mathcal{S}^2} W_\mathrm{e}(\mathbf{y}_0, \boldsymbol{\omega})\, L_\mathrm{i}(\mathbf{y}_0, \boldsymbol{\omega})\, |\cos\theta|\, \mathrm{d}\sigma(\omega), \tag{5.5}$$

this is usually not a particularly good choice. The reason is that for the standard camera models typically used in computer graphics, the contribution of each point on the aperture across all measurements does not vary much. Therefore, we use the simpler approach of sampling the initial vertex of the eye path with uniform probability across the aperture.

Equation 5.4 has a form similar to that of the measurement equation (Equation 2.16), except that rather than measuring sensor response to incident radiance, it measures the "response" of the emitted radiance function to incident importance. The single number, which is the result, is given by

$$I^* = \int_{\mathcal{S}^2} L_\mathrm{e}(\mathbf{x}_0, \boldsymbol{\omega})\, W_\mathrm{i}(\mathbf{x}_0, \boldsymbol{\omega})\, |\cos\theta|\, \mathrm{d}\sigma(\omega), \tag{5.6}$$

and is a new type of measurement, which we will call an *adjoint measurement*. Adjoint measurements are useful because they allow us to implement importance sampling according Equation 5.4. To do so, a single adjoint measurement will not suffice, and therefore it will be necessary to define a number of adjoint measurements, I^*_j, each with their own L^j_e for $j = 1, 2, \ldots, m$. The functions L^j_e can be found by decomposing L_e in various way, as discussed below. Note that this is essentially the opposite situation compared to regular measurements, since in that case we started with the functions W^j_e and formed W_e by combining those, whereas with adjoint measurements we start with L_e and extract the L^j_e's.

An infinite number of ways of decomposing L_e exist, each with their own advantages and drawbacks. The perhaps simplest way of finding the L^j_e's is by decomposing L_e based on the objects in the scene. If a scene has m discrete light sources, we will define L^i_e to be equal to L_e on the ith light source and zero elsewhere, so that

$$L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}) = L^1_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}) + L^2_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}) + \ldots + L^m_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}). \tag{5.7}$$

The results of this decomposition are the adjoint measurements $I_1^*, I_2^*, \ldots, I_m^*$, which tells us the relative importance of each light source. The adjoint measurements can then be used as discrete probabilities,[1] so that rather than choosing the initial vertex in the light path uniformly across the light sources or based on emitted power, the first vertex is chosen (approximately) according to both radiance *and* importance (Equation 5.4). In this way, light sources that contribute light that is actually visible to the sensors will be sampled with higher density, and light sources that are invisible will be ignored no matter how bright they are. Once the initial light source has been identified, a random position can be chosen across its surface area (or inside its volume, in the case of volumetric light sources).

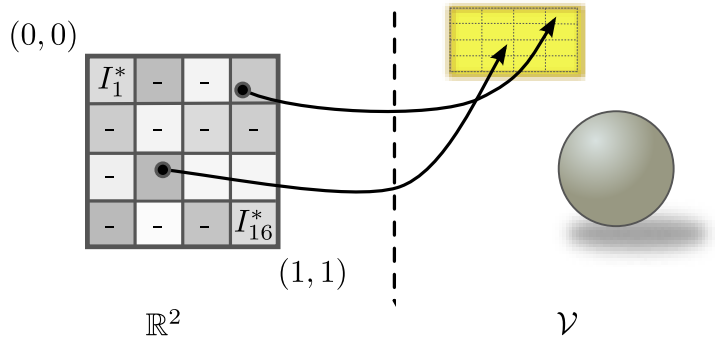
The downside to the above approach is that its effectiveness depends on how the light sources in the scene are organized. In the worst case, if a scene only has a single light source, then the above procedure will be of no benefit at all. To make matters worse, if the scene has large area/volume light sources, then it should be expected that Equation 5.4 could vary considerably across the light sources. Unfortunately, the above procedure does not take this into account, because Equation 5.4 is approximated as a piecewise constant function.

In order to avoid these problems and create an importance sampling procedure that provides a better match for Equation 5.4, we have instead opted to use a method based on warping the random numbers used to find the first vertex in the light path. The idea, which is illustrated in Figure 5.3, is to start out with a simply way of mapping a pair of random numbers in $[0; 1]^2$ to a point on the surface of a light source. We will assume that such a procedure, $F : \mathbb{R}^2 \mapsto \partial\mathcal{V}$, exists and that it is injective. If none exists, constructing one is trivial. For instance, if the scene has m light sources, simply pick one randomly (with probability $1/m$) and then pick a random position on its surface (with probability $1/A_i$, where A_i is the surface area of the chosen light source). This can be done using two random numbers. The functions L_{e}^j can now be defined by subdividing the unit square into smaller domains of equal size. Each domain has an associated L_{e}^j, which is equal to L_{e} across the part of the surface area of $\partial\mathcal{V}$ that this domain maps to and zero elsewhere. The resulting L_{e}^j's must then also satisfy Equation 5.7.

This procedure has the advantage of being independent of how the scene is organized into objects and also solves the problem of variation in Equation 5.4 across area light sources, assuming the resolution is fine enough to capture these variations. We typically divide the unit square into 64×64 domains. We set the actual number of horizontal subdivisions to an integer multiple of the

[1] To ensure unbiasedness, it is necessary to add a small epsilon value to the adjoint measurements, so that there is a non-zero probability of sampling any light source. This is necessary, since the measurements are not evaluated exactly, but are Monte Carlo estimates.

Figure 5.3: The traditional way of finding the initial vertex in a light path is based on mapping a uniform random number in $[0;1]^2$ (left) to a position on the light sources (right). This mapping is usually uniform with respect to area or radiant exitance, which has the disadvantage that importance is not taken into account. To avoid this problem, we first perform a set of adjoint measurements, $I_1^*, I_2^*, \ldots$, to discover which parts of the light sources that matter most when solving the light transport problem. As shown, this is done by subdividing $[0;1]^2$ into smaller regions, and computing how important these regions are. Based on the resulting measurements, we modify the original sampling procedure so that positions on light sources that will tend to lead to light paths that are visible to the sensors will get sampled more often. This can be done by first sampling a region with probability proportional to I_j^* and then picking a random position within that region.



number of light sources, since this ensures that a domain will not cover multiple light sources. The resulting image, which we will call the *adjoint image* to differentiate it from the regular image formed by the normal measurements, is essentially the scene seen by the light sources illuminated by the sensors.

The measurements necessary to create the adjoint image can be computed as part of the Metropolis particle sampling procedure used to create the environment maps with incident radiance/importance described in the next section. This is possible, since the Metropolis method works by sampling paths connecting light sources and sensors. To make this work it should also be possible to perform the inverse mapping, $F^{-1} : \partial\mathcal{V} \mapsto \mathbb{R}^2$. That is, given a position on a light source $\mathbf{x}_s \in \partial\mathcal{V}$, it should be possible to find $F^{-1}(\mathbf{x}_s) \in \mathbb{R}^2$. This is necessary in order to implement multiple importance sampling, so that if an eye path randomly intersects a light source, the probability of starting a light path at that point can be computed. Also, to ensure that the adjoint image does not take too long to converge, the resolution should be significantly smaller than

the final image.

Once the adjoint image has been computed, it is possible to sample the initial vertices of the light paths approximately according to Equation 5.4. To do so we start out with a pair of uniform random numbers in $[0;1]^2$ as usual. However, rather than using these numbers directly as input to F, their distribution is first "warped" according to the adjoint image, before being used to find $\mathbf{x}_s$. Warping the numbers can be done by importance sampling using the inversion method described in Section 3.4. The end result when sampling this way is that the density now depends on both the radiance exitance and on how much of that light energy that ends up at the sensors. This technique can also be extended to volumetric light sources, though we have not implemented this. The main difference is that the adjoint measurement are now arranged in a 3D array, and that a triplet of random numbers in $[0;1]^3$ needs to be warped.

5.2.2 Particle Sampling

The equilibrium radiance distribution and the equilibrium importance distribution can be represented in an unbiased way using a set of weighted rays. Essentially, these weighted rays form a discrete approximation of either $L_\mathrm{i}(\mathbf{x},\boldsymbol{\omega})$ or $W_\mathrm{i}(\mathbf{x},\boldsymbol{\omega})$. It is also possible to represent their exitant counterparts, $L_\mathrm{o}(\mathbf{x},\boldsymbol{\omega})$ and $W_\mathrm{o}(\mathbf{x},\boldsymbol{\omega})$, in this way, though this is less frequently useful. Such approximations form the basis of many global illumination algorithms, such as algorithms based on virtual point light sources and photon mapping based methods.

We use the formulation of particle tracing given by Veach [1997, pg. 121]. A representation of $L_\mathrm{i}(\mathbf{x},\boldsymbol{\omega})$ can be formed by tracing a random walk comprised of N steps in the scene. The result is a set of N rays, $\mathbf{r}_i = (\mathbf{x}_i, \boldsymbol{\omega}_i)$, each with a weight α_i. In order for these weighted rays, $(\mathbf{r}_i, \alpha_i)$, to form an unbiased representation of the equilibrium radiance distribution, the weights must satisfy

$$\mathrm{E}\left[\sum_{i=1}^{N} \alpha_i \, W_\mathrm{e}(\mathbf{x}_i, \boldsymbol{\omega}_i)\right] = \langle W_\mathrm{e}, L_\mathrm{i} \rangle, \tag{5.8}$$

for any W_e. Exactly analogous, the equilibrium importance distribution can by represented by a set of N weighted rays, $(\mathbf{r}_i, \beta_i)$, which is an unbiased approximation of $W_\mathrm{i}(\mathbf{x},\boldsymbol{\omega})$ if

$$\mathrm{E}\left[\sum_{i=1}^{N} \beta_i \, L_\mathrm{e}(\mathbf{x}_i, \boldsymbol{\omega}_i)\right] = \langle L_\mathrm{e}, W_\mathrm{i} \rangle \tag{5.9}$$

is satisfied for any L_e. Another way to express this is using a pair of joint density function, $p_{L_\mathrm{i}}(\alpha, \mathbf{r})$ and $p_{W_\mathrm{i}}(\beta, \mathbf{r})$, that describe the distribution of photons and

importons and their weights. For Equation 5.8 and Equation 5.9 to be satisfied, these joint density functions must fulfill

$$\int_{\mathbb{R}} \alpha\, p_{L_\mathrm{i}}(\alpha, \mathbf{r})\, \mathrm{d}\alpha = L_\mathrm{i}(\mathbf{r}) \quad \text{and} \quad \int_{\mathbb{R}} \beta\, p_{W_\mathrm{i}}(\beta, \mathbf{r})\, \mathrm{d}\beta = W_\mathrm{i}(\mathbf{r}). \tag{5.10}$$

Equations 5.10 describe conditions that must apply in order for the particles to form unbiased representations of radiance or importance, but they do not tell us how to generate the particles in the first place. In order to generate the particles we need an algorithm that can sample the rays and compute the weights. Some of these algorithms are discussed in the following.

The differences between how these algorithms sample rays cause the distribution of the of the generated particles to differ. This can have important consequences when the particles are actually used, since some distributions are more suitable than others depending on what the intended use of the particles are. Another important difference between the algorithms is that some algorithms have the advantage that they can sample radiance and importance simultaneously, while others sample photons and importons with essentially independent simulations.

Classical Particle Sampling

The traditional way of sampling photons in computer graphics is based on recursively expanding the equation of transfer (Equation 4.5) with boundary conditions. This operation is identical to parts one and two of the local path sampling procedure (see Section 4.6.1), and will not be repeated here.

The local path sampling procedure yields a set of rays. In order to compute their weights, we will first simplify notation by writing emission as a product of a positional part and a directional part,

$$L_\mathrm{e}(\mathbf{y}_0 \to \mathbf{y}_1) = L_\mathrm{e}^{(0)}(\mathbf{y}_0)\, L_\mathrm{e}^{(1)}(\mathbf{y}_0 \to \mathbf{y}_1).$$

The directional part can then be written as a special kind of "emission" scattering kernel,

$$f_\mathrm{k}^*(\mathbf{y}_{-1} \to \mathbf{y}_0 \to \mathbf{y}_1) \equiv L_\mathrm{e}^{(1)}(\mathbf{y}_0 \to \mathbf{y}_1),$$

where the vertex $\mathbf{y}_{-1}$ can be thought of as a special virtual vertex that is the source of all radiance in the scene.

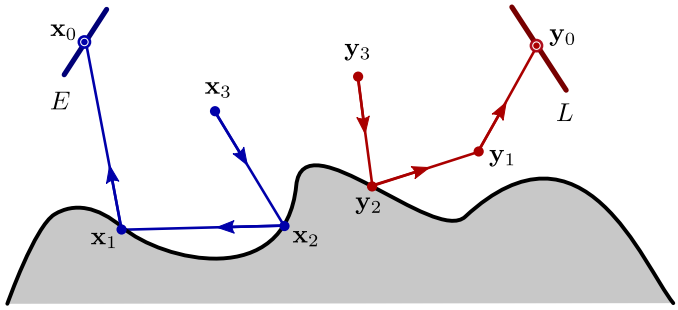
The weights associated with the rays of a particular path $\bar{\mathbf{y}} = \mathbf{y}_0\mathbf{y}_1 \ldots$ are then

given by the expression

$$\alpha_i = \frac{L_{\mathrm{e}}^{(0)}(\mathbf{y}_0)}{p_0(\mathbf{y}_0)} \prod_{j=0}^{i} \frac{f_{\mathrm{k}}^*(\mathbf{y}_{j-1} \to \mathbf{y}_j \to \mathbf{y}_{j+1})\, G(\mathbf{y}_j \leftrightarrow \mathbf{y}_{j+1})}{p_{j+1}(\mathbf{y}_j \to \mathbf{y}_{j+1})}, \tag{5.11}$$

where f_{k}^* is adjoint scattering kernel (adjoint BSDF, phase function, or radiance emission kernel) and G is the generalized geometry term. The PDF, $p_{j+1}(\mathbf{y}_j \to \mathbf{y}_{j+1})$, gives the probability density with respect to area/volume of sampling $\mathbf{y}_{j+1}$ as the next vertex given that the current vertex is $\mathbf{y}_j$. This function includes the probability of sampling the kernel at $\mathbf{y}_j$, the probability associated with propagation step in case of participating media, and the survival probability if Russian roulette is used.

Figure 5.4: The traditional way of sampling particles to represent radiance or importance is based on recursively expanding the equation of transfer or its adjoint version. As shown, this is done by creating random walks starting at the sensor (E) or light source (L). Importons (blue) and photons (red) are stored at each vertex along the resulting paths along with their weights. Note that the arrows point in the opposite direction of the flow of importance/radiance, since the particles represent incident quantities.



This procedure is illustrated in Figure 5.4. As an example, the ith ray of the light path is $\mathbf{r}_i = (\mathbf{y}_{j+1}, \boldsymbol{\omega}_{\mathbf{y}_{j+1} \to \mathbf{y}_j})$ and the ith photon is $(\mathbf{r}_i, \alpha_i)$. Note that directions are opposite to the direction of light flow, since the particles represent incident radiance.

An almost identical procedure can be used to generate importons. First decompose the flux responsivity function into positional and directional parts,

$$W_{\mathrm{e}}(\mathbf{x}_0 \to \mathbf{x}_1) = W_{\mathrm{e}}^{(0)}(\mathbf{x}_0)\, W_{\mathrm{e}}^{(1)}(\mathbf{x}_0 \to \mathbf{x}_1).$$

Then write the directional part as special kind of scattering kernel,

$$f_{\mathrm{k}}(\mathbf{x}_{-1} \to \mathbf{x}_0 \to \mathbf{x}_1) \equiv W_{\mathrm{e}}^{(1)}(\mathbf{x}_0 \to \mathbf{x}_1),$$

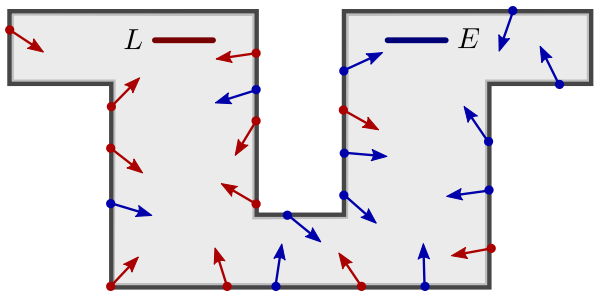
where $\mathbf{x}_{-1}$ is a virtual vertex which can be thought of as the source of all importance. The weights for the rays in the path $\mathbf{x}_0\mathbf{x}_1\ldots$ are then given by

$$\beta_i = \frac{W_{\mathrm{e}}^{(0)}(\mathbf{x}_0)}{p_0(\mathbf{x}_0)} \prod_{j=0}^{i} \frac{f_{\mathrm{k}}(\mathbf{x}_{j-1}\to\mathbf{x}_j\to\mathbf{x}_{j+1})\, G(\mathbf{x}_j\leftrightarrow\mathbf{x}_{j+1})}{p_{j+1}(\mathbf{x}_j\to\mathbf{x}_{j+1})}, \tag{5.12}$$

where f_{k} is the regular scattering kernel (BSDF, phase function, or importance emission kernel). The ith ray $\mathbf{r}_i = (\mathbf{x}_{j+1}, \boldsymbol{\omega}_{\mathbf{x}_{j+1}\to\mathbf{x}_j})$ yields an importon $(\mathbf{r}_i, \beta_i)$ and now the directions point toward the sensor, since the particles represent incident importance.

Sampling using the above procedure generates particles with particular distributions, $p_{L_{\mathrm{i}}}(\alpha, \mathbf{r})$ or $p_{W_{\mathrm{i}}}(\beta, \mathbf{r})$. To investigate this further, consider how the probability density functions in Equation 5.11 and Equation 5.12 influence the weights of the particles. When sampling paths, it is common to make the PDF in the denominator proportional to the corresponding term in the numerator, since this decreases the variance of the weights. In the special case that importance sampling can be carried out exactly, the weights of all the resulting particles will be constant; i.e., all particles will have the exact same weight. Note that even if importance sampling is only nearly exact, this will still be approximately true.

Figure 5.5: The limitations of classical particle sampling. Classical particle attempts to distribute photons so that their density is proportional to radiance (likewise for importance and importons). Unfortunately this means that the density of photons may be low near the sensors and the density of importons low near the light source, which is problematic if we attempt to use the particles in these regions to estimate L_{i} or W_{i}. As shown in the figure, this happens in scenes where the distribution of radiance and importance is very different.



What this means is that the radiance or importance of the scene is encoded in the particles only by how they are distributed throughout the scene and not

by their weights. For instance, brightly lit parts of the scene will be brighter because more photons are located there, rather than because these parts contain photons with higher weights (similarly for importance/importons). Another way too see this is to consider the joint densities. If the particle weights are constant, the joint densities of the particles can be written

$$p_{L_\mathrm{i}}(\mathbf{r},\alpha) = p_{L_\mathrm{i}}(\mathbf{r})\,\delta(\alpha-\alpha_{L_\mathrm{i}}) \qquad \text{and} \qquad p_{W_\mathrm{i}}(\mathbf{r},\beta) = p_{W_\mathrm{i}}(\mathbf{r})\,\delta(\beta-\beta_{W_\mathrm{i}}), \quad (5.13)$$

where α_{L_i} is the weight of the photons and β_{W_i} is the weight of the importons. If these equations are substituted back into Equations 5.10 we get (in the radiance case)

$$\int_{\mathbb{R}} \alpha\, p_{L_\mathrm{i}}(\mathbf{r})\,\delta(\alpha-\alpha_{L_\mathrm{i}})\;\mathrm{d}\alpha = \alpha_{L_\mathrm{i}}\, p_{L_\mathrm{i}}(\mathbf{r}) = L_\mathrm{i}(\mathbf{r}),$$

and similarly for importance. This shows that $p_{L_\mathrm{i}}(\mathbf{r}) \propto L_\mathrm{i}(\mathbf{r})$ and $p_{W_\mathrm{i}}(\mathbf{r}) \propto W_\mathrm{i}(\mathbf{r})$ (and that the constants of proportionality are α_{L_i} and β_{W_i}).

Constant, or low-variance, weights can be an advantage for some algorithms, such as those based on density estimation. However, for other algorithms near-constant weights can be a disadvantage, since this means that particles are sampled without considering the corresponding adjoint quantity. For instance, if the sensors and light sources are separated by difficult geometry, very little importance may reach the light sources and very little radiance may reach the sensors. As shown in Figure 5.5, this means that only few importons will be found near the light sources and few photons near the sensors. Unfortunately, this is exactly where we need them the most. Recall that in the algorithm described earlier we attempt to guide eye paths to light sources using the incident radiance and guide light paths to the sensor using the incident importance. If no photons are found near the sensor and no importons near the light source, it will be difficult to accurately estimate the relevant incident quantity and this strategy will not be possible. As a consequence, this type of particle sampling is not well suited for the proposed algorithm.

Bidirectional Particle Sampling

To overcome the aforementioned problem of unsuitable particle distributions, it will be necessary to develop a new way of sampling particles. Such an algorithm should generate particles with marginal densities, $p_{L_\mathrm{i}}(\mathbf{r})$ and $p_{W_\mathrm{i}}(\mathbf{r})$, such that the density of particles in a given region of $\mathbb{R}^3 \times \mathcal{S}^2$ depends on how relevant this region is to the problem being solved.

In order to apply this strategy it will be necessary to define exactly what relevancy means in this context. Recall that the problem we are attempting to solve

is the light transport problem given by Equation 5.1. This problem is formulated as an integral over all paths, which implies that any region of $\mathbb{R}^3 \times \mathcal{S}^2$ could potentially be relevant (this is essentially the "global" nature of global illumination). However, not all paths contribute equally as defined by the measurement contribution function, $f(\bar{\mathbf{x}})$, and consequently not all parts of $\mathbb{R}^3 \times \mathcal{S}^2$ will be equally important. It follows that one natural way of defining the relevancy of a subset $\mathbb{R}^3 \times \mathcal{S}^2$ of the scene could be by basing it on the contributions of the paths that pass through that region.

Two definitions are required to make this notion more precise. First, let $\Omega_{L_\mathrm{i}}(\mathbf{r})$ be the set of paths that include the ray $\mathbf{r} = (\mathbf{x}, \boldsymbol{\omega})$, where it is understood that $\mathbf{x}$ is a vertex in the path and $\boldsymbol{\omega}$ is the outgoing direction of the path from that vertex in the direction of the light source. Similarly, $\Omega_{W_\mathrm{i}}(\mathbf{r})$ is the set of paths that include $\mathbf{r} = (\mathbf{x}, \boldsymbol{\omega})$, but with the important difference that $\boldsymbol{\omega}$ points toward the sensor. Suitable particle densities are then given by

$$p_{L_\mathrm{i}}(\mathbf{r}) \propto \int_{\Omega_{L_\mathrm{i}}(\mathbf{r})} f(\bar{\mathbf{x}})\, \mathrm{d}\mu(\bar{\mathbf{x}}) \qquad \text{and} \qquad p_{W_\mathrm{i}}(\mathbf{r}) \propto \int_{\Omega_{W_\mathrm{i}}(\mathbf{r})} f(\bar{\mathbf{x}})\, \mathrm{d}\mu(\bar{\mathbf{x}}). \quad (5.14)$$

Basically, using these densities ensures that regions of $\mathbb{R}^3 \times \mathcal{S}^2$ that "matter" when solving Equation 5.1 get sampled more densely, since the distributions of particles are now based on both radiance and importance.

The discussion above shows that desirable particle distributions should be based on both L_i and W_i, but it does not explain how to actually generate particles that follow these distributions. The simplest way to do this is to use classical particle sampling combined with a resampling step. The idea is as follows: first generate a set photons and a set of importons using the classical particle sampling procedure described above. The result is two sets of particles; one distributed approximately according to radiance and one according to importance. Based on these particles it is now possible to estimate $L_\mathrm{i}(\mathbf{r})$ and $W_\mathrm{i}(\mathbf{r})$ anywhere in the scene, e.g. using some form of density estimation. The second part of the procedure is a resampling step to "thin out" oversampled regions. For each particle it is randomly decided whether to keep it in the set. This can be done using a variation of Russian roulette, where the probability of survival, p, depends on the adjoint quantity, which can be estimated using the other set of particles. For instance, each photon, $(\mathbf{r}_i, \alpha_i)$ with $\mathbf{r}_i = (\mathbf{x}_i, \boldsymbol{\omega}_i)$, is terminated with a probability that depends on how much the incident importance at $\mathbf{x}_i$ from all directions is scattered in the direction of $\boldsymbol{\omega}_i$. If the photon survives, the weight of the photon becomes α_i / p to ensure that the particles remain an unbiased approximation of L_i. Importons can be resampled in the same way.

The above procedure produces particles distributed approximately according to the product of radiance and importance. This is done by reducing the number

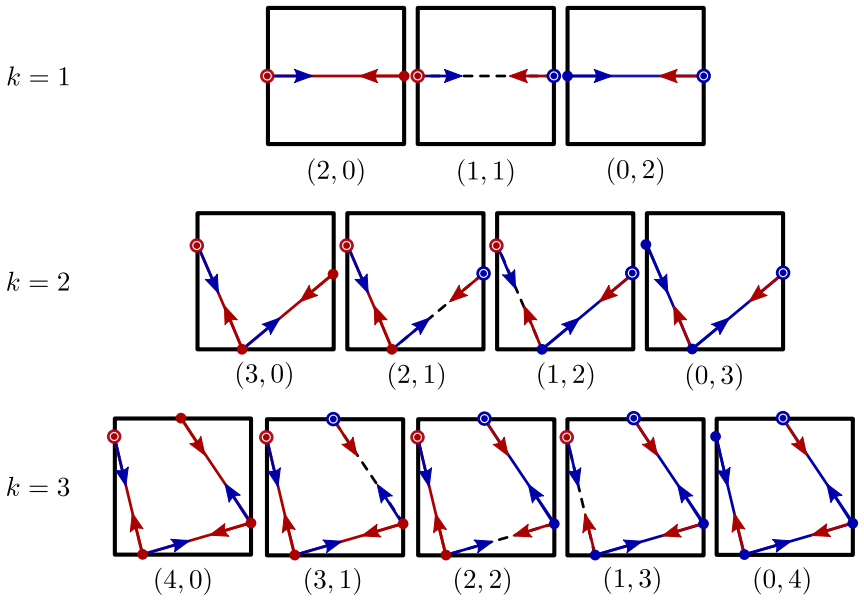
of particles in oversampled regions and adjusting the weights of the surviving particles. For instance, importons in regions with high radiance will be more likely to survive and thus be more numerous, so to compensate their weights must be made smaller. The downside to this approach is that in some cases many particles will have to be terminated, and these particles are essentially wasted. This happens if the distribution of radiance and importance in a scene is very different, since then few particles will reach regions near the source of the corresponding adjoint quantity. As a consequence, a large number of particles will have to be generated and eventually terminated in order to reach a sufficient sample density in those regions. To make matters worse, using Russian roulette with very low survival probabilities can increase the variance of weights. This is obviously very inefficient, and shows that it would be better if it was possible to directly sample particles distributed according to Equation 5.14, rather than having to resort to an intermediary distribution and resampling step.

Unfortunately, directly sampling particles distributed according to Equation 5.14 is not possible, since that requires sampling paths with density proportional to $f(\bar{\mathbf{x}})$ (recall that sampling paths with distributions that closer match the measurement contribution function is exactly the problem we are trying to solve in the first place). However, it may still be possible to generate particles with distributions along the lines of Equation 5.14 by using the full framework provided by local path sampling. Classical particle tracing, as outlined above, is essentially a unidirectional approach, similar to pure path tracing or light tracing. E.g., photons are only sampled using paths starting on the light sources, and therefore the particle density may be low near the sensor. In order to avoid this, photons can also be sampled from paths starting on the sensors and ending on the light sources, which should cause the photon density near the sensors to increase. Similarly, importons could be sampled using paths starting on the light sources, which should cause the importon density to increase near light sources. An additional advantage of this is that it is sometimes easier to find a path connecting lights and sensors if the path is started from the sensors, so using a sampling strategy based on performing random walks in both directions should be beneficial. This is closely related to the discussion of the advantages of bidirectional path tracing over path tracing and light tracing in the previous chapter, and suggests that particle sampling based on a bidirectional method could be advantageous.

In order to carry out bidirectional particle sampling it is necessary to develop a way of generating the particles and a method of computing their weights. This must be done in such a way that the resulting joint densities, $p_{L_\mathrm{i}}(\alpha, \mathbf{r})$ and $p_{W_\mathrm{i}}(\beta, \mathbf{r})$, still satisfy Equations 5.10. I.e., the photons/importons should still form an unbiased representations of radiance/importance in the scene even if the resulting distributions are now different.

Bidirectional particle sampling is based on sampling complete paths that connect the sensor and light sources. Once a path has been created, photons and importons can be extracted from the vertices along the path. This procedure is illustrated in Figure 5.6 for paths of lengths one through three. Since a photon and importon is generated at each vertex (except the first and last vertices), a path of length k produces up to k photons and k importons. These paths are created by first performing two random walks, one starting from the sensor and one from the light source, and then connecting the endpoints of two random walks. Note that unlike bidirectional path tracing, a sample is here a single path and not a whole family of paths (compare to Figure 4.4).

Figure 5.6: Bidirectional particle sampling works by generating full paths that connect the sensor and light source and extracting photons and importons from the vertices along the paths. Paths are created using a pair of random walks from the sensor and light sources whose endpoints are connected by tracing a shadow feeler (dashed lines). The numbers underneath each figure indicate how many vertices the light path and eye path contain using standard (s, t) notation. Note that in case of a $(s, 0)$-method or a $(0, t)$-method, one of these random walks will be empty.



In order to find the weight of a particle, it is necessary to compute how much radiance or importance is transported along the path to the particle and then divide this value by the probability of generating the path segment. The incident radiance at a given vertex is simply the product of geometry terms and scattering

kernels. Computing the probability of the particle is more involved and depends on the strategy used to sample the paths. It should be clear that simply sampling paths of given fixed length will not yield the correct result. E.g. if photons are only extracted from paths of length $k = 2$ only direct lighting and the first indirect bounce will be accounted for. Therefore, there should be a non-zero probability of sampling paths of any length so that radiance (or importance) that has scattered any number of times is properly accounted for.

The first task is therefore to randomly choose the length of the path. We use the following probabilities for sampling paths of length k,

$$p_k = \frac{1}{2^k}.$$

Using these probabilities ensures that there is a chance of sampling paths of any length. It also favors shorter paths, since these are usually the most important. Sampling photons (or importons) in this way means the probability of generating a given particle depends on how many scattering events it is away from the light source (or sensor). For instance, photons corresponding to the first indirect bounce ($i = 2$) are only generated if the sampled path has length $k \geq 2$. This means that the probability gets an extra factor of

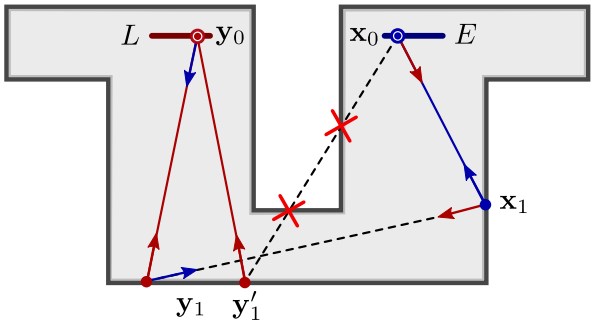
$$q_i = \sum_{k=i}^{\infty} p_k = 1 - \sum_{k=1}^{i-1} p_k,$$

to compensate. Therefore particles corresponding to direct illumination get an extra factor $q_1 = 1$, since these are always generated no matter the value of k (refer to Figure 5.6). On the other hand, particles corresponding to the nth indirect bounce get an additional factor of q_{n+1}.

Once the length of the path has been determined, the number of vertices in the light path, s, versus the number in the eye path, t, must be decided. Since we have no a priori knowledge of which methods are best for a given scene, we simply pick randomly among the $k + 2$ variations. I.e., the length of the light path, s, is selected with uniform probability among $[0, 1, \ldots, k + 1]$, and the number of vertices in the eye path is then given by $t = k - s + 1$.

Once we know the desired length, we perform the two random walks with a maximum of s or t vertices in each. Figure 5.7 shows two paths generated this way and the resulting particles. We will use these two paths to demonstrate how to compute the weights of the particles. The first path is $\bar{\mathbf{z}} = \mathbf{y}_0\mathbf{y}_1\mathbf{x}_1\mathbf{x}_0$, which was generated using a $(2, 2)$-method (two vertices in the light path and two vertices in the eye path). For this path the path segment between the connecting vertices, $\mathbf{y}_1$ and $\mathbf{x}_1$, is not blocked, so sampling this path results in

Figure 5.7: Two paths generated during bidirectional particle sampling. The first path has length three and contributes the full compliment of three photons and three importons. The second path has length two and the ray segment between the connecting vertices, $\mathbf{y}_1'$ and $\mathbf{x}_0$, is blocked. Since the path is blocked it contributes fewer particles (in this case only one photon and no importons).



three photons and three importons. As we will see later, sometimes a path of length k will result in fewer than k photons and k importons.

Consider computing the weight of the photon at $\mathbf{x}_1$ (the procedure for computing the weights of the importons is exactly analogues). To find the weight we need to first find the incident radiance along the subpath of $\mathbf{y}_0\mathbf{y}_1\mathbf{x}_1$,

$$f_L(\mathbf{y}_0\mathbf{y}_1\mathbf{x}_1) = L_e(\mathbf{y}_0 \rightarrow \mathbf{y}_1) G(\mathbf{y}_0 \leftrightarrow \mathbf{y}_1) f_k^*(\mathbf{y}_0 \rightarrow \mathbf{y}_1 \rightarrow \mathbf{x}_1) G(\mathbf{y}_1 \leftrightarrow \mathbf{x}_1).$$

The next step is to compute the path space probability of sampling the subpath $\mathbf{y}_0\mathbf{y}_1\mathbf{x}_1$ as part of $\bar{\mathbf{z}}$. Since $\bar{\mathbf{z}}$ has length three, this can be done in up to five different ways, corresponding to how many vertices there are in the light subpath versus the eye subpath. The probabilities are given by

$$\begin{aligned}
p_{4,0} &= p_L(\mathbf{y}_0)\, p_L(\mathbf{y}_0 \rightarrow \mathbf{y}_1)\, p_L(\mathbf{y}_1 \rightarrow \mathbf{x}_1) \\
p_{3,1} &= p_L(\mathbf{y}_0)\, p_L(\mathbf{y}_0 \rightarrow \mathbf{y}_1)\, p_L(\mathbf{y}_1 \rightarrow \mathbf{x}_1) \\
p_{2,2} &= p_L(\mathbf{y}_0)\, p_L(\mathbf{y}_0 \rightarrow \mathbf{y}_1)\, p_W(\mathbf{x}_0)\, p_W(\mathbf{x}_0 \rightarrow \mathbf{x}_1) \\
p_{1,3} &= p_L(\mathbf{y}_0)\, p_W(\mathbf{x}_0)\, p_W(\mathbf{x}_0 \rightarrow \mathbf{x}_1)\, p_W(\mathbf{x}_1 \rightarrow \mathbf{y}_1) \\
p_{0,4} &= p_W(\mathbf{x}_0)\, p_W(\mathbf{x}_0 \rightarrow \mathbf{x}_1)\, p_W(\mathbf{x}_1 \rightarrow \mathbf{y}_1)\, p_W(\mathbf{y}_1 \rightarrow \mathbf{y}_0).
\end{aligned}$$

Note that these are not the same probabilities used in bidirectional path tracing (see Section 4.7.3), since they are probabilities for sampling the path segment, $\mathbf{y}_0\mathbf{y}_1\mathbf{x}_1$, rather than the full path. For instance, $p_{4,0}$ and $p_{3,1}$ are the same, since these probabilities are independent of how the last path segment of $\bar{\mathbf{z}}$ was sampled.

We now have all the information needed for computing the weight of the photon at $\mathbf{x}_1$. If we use multiple importance sampling, the weight can be written as

$$\alpha_2 = \frac{5\, p_{2,2}^2}{p_{4,0}^2 + p_{3,1}^2 + p_{2,2}^2 + p_{1,3}^2 + p_{0,4}^2} \frac{f_L(\mathbf{y}_0\mathbf{y}_1\mathbf{x}_1)}{p_{2,2}\, q_2}.$$

The multiplication by $k+2 = 5$ is necessary, since we are only using one of the $k+2$ methods in each sample. We used the power heuristic with $\beta = 2$, which means that the multiple importance weight of the sample is linear in probability.

In the above example we computed the weight of a photon in a complete path, i.e. a path that connected the sensor and light source. Often the sampled path does not connect the sensor and light sources. This can happen because the generated random walks are shorter than desired (s or t vertices respectively), which is caused by the paths escaping the scene or because the paths are terminated early using Russian roulette. It can also happen because the connecting vertices in the random walks are not mutually visible (only for $s \geq 1$ and $t \geq 1$) Finally, it can happen because $(s, 0)$-methods fail to randomly hit the sensor or because $(0, t)$-methods fail to hit a light source. All these cases have to considered when computing weights.

Figure 5.7 also shows a path, $\mathbf{y}_0\mathbf{y}_1'\mathbf{x}_0$, where the connecting vertices are blocked. This path only contributes a single photon and no importons (the photon at $\mathbf{x}_0$ must have weight zero, since the geometry term, $G(\mathbf{y}_1' \leftrightarrow \mathbf{x}_0)$, is zero due to the visibility function). Similarly, the weights for the importons are also zero.

In order to compute weight of the photon at $\mathbf{y}_1'$ we need to find the probabilities. The $k+2$ probabilities are given by

$$\begin{aligned}
p_{3,0} &= p_L(\mathbf{y}_0)\, p_L(\mathbf{y}_0 \rightarrow \mathbf{y}_1') \\
p_{2,1} &= p_L(\mathbf{y}_0)\, p_L(\mathbf{y}_0 \rightarrow \mathbf{y}_1') \\
p_{1,2} &= p_L(\mathbf{y}_0)\, p_W(\mathbf{x}_0)\, p_W(\mathbf{x}_0 \rightarrow \mathbf{y}_1') = 0 \\
p_{0,3} &= p_W(\mathbf{x}_0)\, p_W(\mathbf{x}_0 \rightarrow \mathbf{y}_1')\, p_W(\mathbf{y}_1' \rightarrow \mathbf{y}_0) = 0.
\end{aligned}$$

However, note that because the ray segment between $\mathbf{y}_1'$ and $\mathbf{x}_0$ is blocked some of these probabilities are now zero. In particular, the term $p_W(\mathbf{x}_0 \rightarrow \mathbf{y}_1')$ is zero, since the conversion from probability measured with respect to solid angle to probability measured with respect to area/volume also includes a visibility term. The weight is thus

$$\alpha_1 = \frac{4\, p_{2,1}^2}{p_{3,0}^2 + p_{2,1}^2 + p_{1,2}^2 + p_{0,3}^2} \frac{f_L(\mathbf{y}_0\mathbf{y}_1')}{p_{2,1}\, q_1} = \frac{2\, f_L(\mathbf{y}_0\mathbf{y}_1')}{p_{2,1}\, q_1}.$$

What this means is that bidirectional particle sampling is reduced to classical particle sampling in the special cases mentioned above. The extra factor of two

is necessary to compensate for the fact that half of the method do not produce any photons.

Metropolis Particle Sampling

Bidirectional particle sampling generates particles with distributions that closer match Equations 5.14 compared to classical particle sampling. This is the case since paths are sampled based on both the distribution of radiance and the distribution of importance in the scene. However, while the distributions of photons and importons may be better, the particles are still not distributed exactly according to Equations 5.14. The reason for this is again that it is not possible to sample paths distributed exactly according to the measurement contribution function using regular local path sampling.

As discussed in Section 4.7, it is possible to use the framework of Metropolis-Hastings in combination with local path sampling. When this strategy is used, it becomes possible to sample paths whose distribution, in the limit, follow the measurement contribution function. The idea is to construct a Markov chain[2] with an appropriate stationary distribution and then evolve this chain using Algorithm 3.2 and a set of predefined mutations strategies.

The stationary distribution of the chain can be chosen to be proportional to the importance weight of a path,

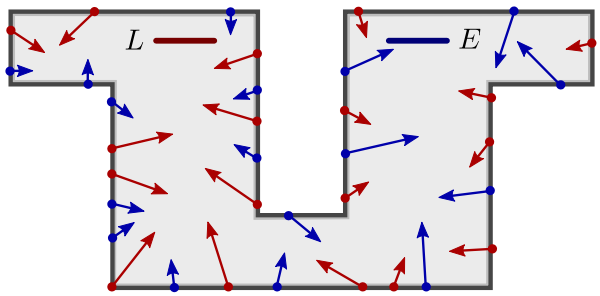
$$\pi^*(\bar{\mathbf{x}}) \propto \frac{f(\bar{\mathbf{x}})}{p(\bar{\mathbf{x}})}, \tag{5.15}$$

where $f(\bar{\mathbf{x}})$ is the measurement contribution function and $p(\bar{\mathbf{x}})$ is probability of sampling the path as described above. Using this stationary distribution has the advantage of placing more samples where they matter most. I.e., paths that are undersampled with local path sampling compared to their contribution will receive more samples, whereas paths with small contributions that are oversampled with local path sampling will receive fewer samples (see Figure 5.8).

Sampling particles from paths based on a Markov chain with stationary distribution given by Equation 5.15 will not always lead to particle distributions that are unbiased representation of their respective quantity. To see why consider a scene, where one of the light sources is completely isolated from the sensor. Since a chain with stationary distribution given by Equation 5.15 will only visit paths where $f(\bar{\mathbf{x}}) > 0$, no particles will be sampled from paths starting on the isolated light source, and the contribution from this light source will therefore

[2]See Section 3.8.2 for further details.

Figure 5.8: Generating particles using Metropolis particle sampling ensures that the density of both photons or importons is high in regions of $\mathbb{R}^3 \times \mathcal{S}^2$ that matter (compare with Figure 5.5). To compensate for the changed density compared to classical particle sampling, the weights of the particles need to be adjusted so that they remain unbiased representations of radiance/importance. In the figure this means that the weights of photons in the brightly lit left part must be increased (illustrated with longer arrows) since there are now relatively fewer and the weights of the photons near the sensor in the right part decreased since there are now more (the situation for importons is exactly the opposite).



be completely missing. This is obviously biased. However, since isolated light sources do not contribute to Equation 5.1, they are of no interest to us. Therefore, not generating any particles to represent these light source is actually an advantage, since time/memory is not wasted generating/storing these particles. Note that if all light sources in a scene are isolated from the sensors, it does not make sense to sample particles using the above stationary distribution; of course, such scenes are of little practical interest.

Ideally, the initial state in Algorithm 3.2 should be chosen according to Equation 5.15, since other choices will lead to startup bias. As discussed in Section 3.8.2, startup bias can be avoided by letting the chain forget its initial state, which can be achieved by ignoring the first k samples. Unfortunately, the number of samples to ignore can be difficult to determine, since it depends on the mixing properties of the chain. A better approach is given in Veach [1997, pg. 339], which is based on a two-stage sampling procedure. First a large number of paths are sampled using regular bidirectional path tracing. Then a smaller number of paths from this population are resampled according to their importance weight. We typically use 65536 paths for the initial population and from this resample 1024 paths according to Equation 5.15. The resampled paths can then be used as the initial states of 1024 chains, which can be evolved

independently across multiple CPU cores, if desired.

Once the initial state of the chain has been determined, the rest of Algorithm 3.2 can be executed. The challenging part of the algorithm is devising a way of proposing tentative samples. As discussed in Section 4.7.4, two different strategies exists for proposing tentative samples (mutating paths): the original strategy by Veach and Guibas [1997], which mutates paths by directly changing their geometries and the newer variation by Kelemen et al. [2002], which operates in the primary sample space (the unit hypercube of random numbers) and mutates paths by perturbing the random numbers used to generate the paths. We have used the latter strategy with the extensions to participating media proposed in Raab et al. [2007], though it should also be possible to use the original strategy by Veach and Guibas.

5.2.3 Clustering

The result of the particle sampling algorithms described above are two sets particles, photons and importons, that present radiance and importance in the scene. These particles are distributed approximately according to Equations 5.14 and therefore have higher sample density where the particles will be needed, unlike the particle distributions that result from classical particle tracing.

While a set of particles can be a relatively compact representation of radiance/importance, they do not lend themselves well to importance sampling. Recall that our goal is to be able to importance sample according to incident radiance/importance, so that light paths can be guided to sensors and eye paths to light sources. While it certainly is possible to estimate these incident quantities directly from the particles, e.g. using some kind of density estimation, this can be very time consuming, since it involves searching for the nearest particles. In addition, once the nearest particles have been identified, they will have to be organized in a way that allows for importance sampling, which can also be time consuming. Since this procedure has to be performed for each vertex when generating paths using local path sampling, this will simply make generating paths too slow (note that estimating the incident quantity will also have to be performed even if the outgoing direction from a given vertex is sampled from the scattering kernel, since this is necessary to facilitate multiple importance sampling).

Rather than using the particles directly, we will instead use the particles to build another data structure, which will be used during rendering and which is better suited for importance sampling. The basic idea is to divide the scene into a set of regions, and store the average incident radiance/importance over those

regions in a way that allows for easy importance sampling. The 5D incident quantities, $L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega})$ or $W_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega})$, are then approximated as piecewise constant with respect to the 3D positional component, whereas the 2D directional component is stored for each region in an image (essentially an environment map) using some suitable parametrization of the sphere. The advantage of this data structure, compared to using the particles directly, is that we only need to determine the current region to find the incident quantities, which is faster than locating a large number particles. Another advantage is that the environment map can be stored in a convenient way that allows for fast importance sampling.

In order to apply the above strategy it is necessary to find a way to divide the scene into regions. Recall that the scene is defined as a finite volume, $\mathcal{V} \subset \mathbb{R}^3$, with associated scattering and emission properties, which in turn is composed of a number of cells with boundary $\partial\mathcal{V}$ and interior $\mathcal{V}_0$. Before presenting the chosen solution, a few observations regarding desirable properties of such decomposition are in order. Firstly, rather than dividing $\mathcal{V}$ into regions of equal size, it is better to let the size of a region depend on how important that part of the scene is to solving Equation 5.1. The reason is that since we intend to store average quantities within each region, a large region will invariably mean a poorer approximation, and therefore small regions are preferable in important parts of the scene. Secondly, rather than directly dividing $\mathcal{V}$ into regions, it is better to handle $\partial\mathcal{V}$ and $\mathcal{V}_0$ separately, so that a given region only contains points from either $\partial\mathcal{V}$ or $\mathcal{V}_0$, but not both sets. The reason for this is that radiance/importance changes discontinuously at boundaries, so having regions that span multiple cells will lead to poor approximations of the incident quantities. This means that the average incident radiance/importance is computed either as surface integral or a volume integral, depending on whether the region belongs to $\partial\mathcal{V}$ or $\mathcal{V}_0$. I.e., if $X \subset \partial\mathcal{V}$ and $Y \subset \mathcal{V}_0$ then

$$L_{\mathrm{i,avg}}(\boldsymbol{\omega}) = \int_X L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}) \, \mathrm{d}A(\mathbf{x}) \qquad \text{and} \qquad L_{\mathrm{i,avg}}(\boldsymbol{\omega}) = \int_Y L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}) \, \mathrm{d}V(\mathbf{x}),$$

and similarly for $W_{\mathrm{i,avg}}(\boldsymbol{\omega})$. Finally, given a point in the scene, $\mathbf{x}_s$ or $\mathbf{x}_v$, it should be possible to quickly determine which region the point belongs to.

A solution to this problem, which fulfills the above requirements, is to define the regions based on a Voronoi decomposition of $\partial\mathcal{V}$ and $\mathcal{V}_0$. To do this it is necessary to first define two point sets, one with points distributed across $\partial\mathcal{V}$ and one with points in $\mathcal{V}_0$, which will be the centers of the regions. It will also be necessary to define a distance function, which will be used to decide which points belong to what region. The region centers should be distributed so that there are more in important parts of the scene, since this will ensure better approximations, as discussed above. We achieve this by using particles generated using Metropolis particle tracing to find the centers in the following way: First, a large number of particles are generated (typically on the order of

10^6) and these are divided into two sets depending on whether they belong to $\partial\mathcal{V}$ or $\mathcal{V}_0$. Then, a k-means clustering [Johnson and Wichern, 2002] is performed on each set, which results in a number of cluster centers (typically on the order of 10^3), which are used directly as the final region centers. Lastly, the two sets of region centers are organized in a kd-tree [Bentley, 1975], so that the nearest region center can be found quickly. This is done for both photons and importons, so the result is four kd-trees with region centers: one for each of surface/volume radiance/importance.

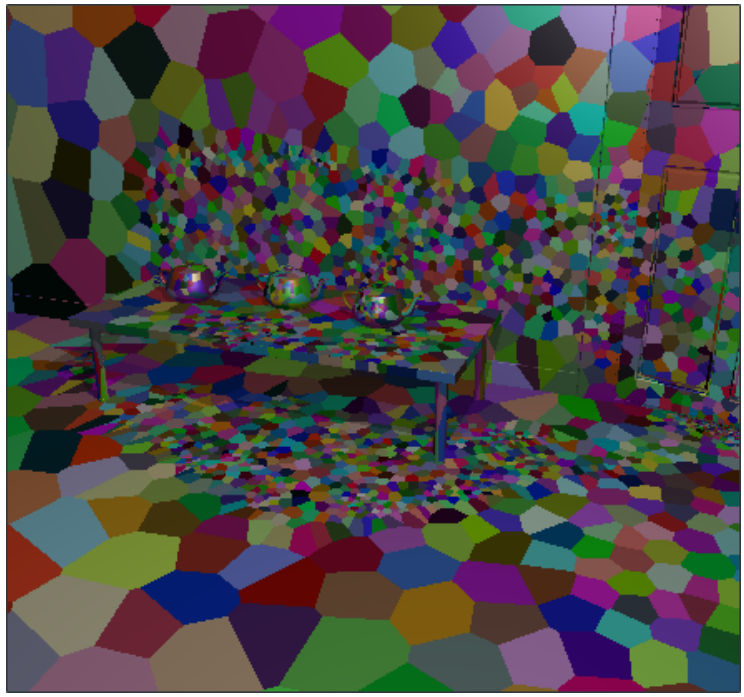
In order to perform k-means clustering it is necessary to define a distance function. The classical choice is to use Euclidean distance to classify points. However, in the surface case ($\partial\mathcal{V}$), this leads to poor results, since points on opposite sides of e.g. a thin wall can end up being clustered together. This is suboptimal, since the incident radiance/importance at these points can be very different, and therefore these points should not be part of the same region. To avoid this a more sophisticated distance function should be used. A distance function which measures actual geodesic distance along the manifold would suffer less from this problem, but would be prohibitively expensive to evaluate. A compromise can be made by using a distance function based on Euclidean distance, but which also uses information about the normal at the region center and at the point on the surface where the particle hit. We used the function

$$\mathrm{dist}(\mathbf{x}_s, \mathbf{x}_c) = \alpha \left|\mathbf{x}_s - \mathbf{x}_c\right| + (1 - \mathbf{n}(\mathbf{x}_s) \cdot \mathbf{n}(\mathbf{x}_c)), \tag{5.16}$$

where $\mathbf{x}_s$ is the position of the particle, $\mathbf{x}_c$ is the region center, and $\mathbf{n}$ is the normal at the given point. The α parameter is necessary to control the influence of the distance part versus the normal part, and depends on the scale of scene. While not completely foolproof, taking curvature into account tends to be significantly better than using Euclidean distance alone, and in addition the function remains fast to evaluate. To actually use it, it is necessary to store the (shading) normal with each particle. It is also necessary to store a normal together with the region center and update this normal as part of the k-means algorithm. An example of the clusters resulting from using this procedure is shown in Figure 5.9.

For the volume case ($\mathcal{V}_0$), the situation is slightly different. As mentioned above, here we want to avoid clustering particles across cell boundaries. We solve this problem by giving each cell a unique identifier and then only clustering particles from the same cell using the ordinary Euclidean distance function. To accomplish this, it is necessary to store the cell identifier with each particle. It is also necessary to store the cell identifier together with the region center, so that it is possible to locate the nearest region center which is also in the same cell when performing the k-means clustering. This too is not completely foolproof, but tends to work well enough in practice for most scenes.

Figure 5.9: The Voronoi regions resulting from clustering the photons shown in Figure 5.13 using the k-means algorithm and the distance function from Equation 5.16. Note how the regions automatically get smaller in the important parts of scene where the photon density is high. This is an advantage, since having smaller regions means that the environment map with incident radiance is computed by averaging over a smaller surface area and therefore is more accurate. The regions resulting from clustering the importons are very similar.



5.2.4 Projection

Once the region centers and their extents have been determined, the environment maps with incident radiance or importance needs to be computed. These environment maps can be computed using the particles that were also used to find the region centers using the k-means algorithm. This is done by taking the set of particles that were assigned to a particular region and adding their weights to the texels in the environment map that corresponds to their incoming directions. Often, too few particles will be assigned to a given region for the environment map with incident radiance/importance to converge properly. Too overcome this problem, more particles can simply be generated. I.e., once the current set of particles have been assigned to their respective regions and

projected, they can be discarded. A new set of particles can then be traced, assigned, and projected, and so on until the environment maps have converged.

Numerous different environment map parametrizations exist. We use a variation of the ordinary latitude-longitude parametrization, where each texel subtends the same solid angle. Compared to the regular latitude-longitude format, this parametrization, which is also known as Peters' Projection, has the advantage that because each texel has the same solid angle, the distribution of samples (particles) across the texels will be more uniform. Since the environment maps represent average incident quantities, they tend to be low-frequency, and therefore need not be high resolution (we use a resolution of 32×16).

Once the environment maps have been computed, it will be necessary transform them into a form suitable for importance sampling, so that directions can be sampled with densities $p(\boldsymbol{\omega}) \propto L_{\mathrm{i}}(\boldsymbol{\omega})$ or $p(\boldsymbol{\omega}) \propto W_{\mathrm{i}}(\boldsymbol{\omega})$. Environment maps are essentially tabulated 2D functions, and they can be sampled using a numerical version of the inversion method described in Section 3.4. Importance sampling such a function with the inversion method is a two step procedure, where first the row is chosen according to a 1D CDF, and then the column is chosen according to another 1D CDF. The CDF for choosing the row is based on the marginal density, which can be found by integrating the environment map along each row of texels. The CDFs for choosing the columns can be computed from the rows of the environment map. Inverting a tabulated CDF can be done using a binary search. This means that to find a texel corresponding to a pair of random numbers, we first perform a binary search to find the row and then another binary search to find the column. Once the texel has been found, a random direction within the solid angle of that texel is chosen. Due to the parametrization used, each texel has the same solid angle $4\pi/n$, where n is the number of texels in the environment map. This means that the final probability becomes

$$p(\boldsymbol{\omega}) = p_{\mathrm{row}}\, p_{\mathrm{column}}\, \frac{n}{4\pi},$$

where p_{row} and p_{column} are the probabilities of sampling the texel.

Memory consumption can be an issue if the scene has many regions. If this is the case, it may not be possible to store all the environment maps in main memory at the same time both during preprocessing and rendering. Memory requirements can be reduced during preprocessing by instead storing the environment maps in a memory mapped file. This can be achieved in the following way: First, generate the *kd*-tree with the region centers as described above and initialize a memory mapped file with empty environment maps. Next, trace particles and sort these according to what region they belong to using the *kd*-tree. Finally, load the environment maps from disk one by one and add the particles from the corresponding region. Once enough particles have been traced so that the

environment maps will have converged, the environment maps can be converted to the CDF form and compressed using the technique described in Lawrence, Rusinkiewicz, and Ramamoorthi [2005], which should ensure that the memory requirements during rendering also become more manageable.

5.3 Rendering Pass

In the previous section we saw how global context could be created using particle sampling. In this section we will discuss how this global context can be used to improve the local path sampling procedure. A summary of the generated global context and its intended use is shown in the following table:

Type	Quantity	Purpose
Adjoint image	I_j^*	Used for sampling the initial vertex in light paths.
Surface radiance tree	$L_{\mathrm{i}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega})$	Used for sampling outgoing directions from vertices, $\mathbf{x}_{\mathrm{s}} \in \partial\mathcal{V}$, in eye paths.
Surface importance tree	$W_{\mathrm{i}}(\mathbf{x}_{\mathrm{s}}, \boldsymbol{\omega})$	Used for sampling outgoing directions from vertices, $\mathbf{x}_{\mathrm{s}} \in \partial\mathcal{V}$, in light paths.
Volume radiance tree	$L_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega})$	Used for sampling outgoing directions from vertices, $\mathbf{x}_{\mathrm{v}} \in \mathcal{V}_0$, in eye paths.
Volume importance tree	$W_{\mathrm{i}}(\mathbf{x}_{\mathrm{v}}, \boldsymbol{\omega})$	Used for sampling outgoing directions from vertices, $\mathbf{x}_{\mathrm{v}} \in \mathcal{V}_0$, in light paths.

5.3.1 Light Source Sampling

The first task when creating the light path is to find the position of the initial vertex in the random walk. Rather than sampling this vertex according to radiant exitance, we sample it using the adjoint image as detailed in Section 5.2.1. This means that the initial vertex, $\mathbf{x}_0$, is sampled with density given approximately by

$$p(\mathbf{x}_0) \propto \int_{\mathcal{S}^2} L_{\mathrm{e}}(\mathbf{x}_0, \boldsymbol{\omega})\, W_{\mathrm{i}}(\mathbf{x}_0, \boldsymbol{\omega})\, |\cos\theta|\, \mathrm{d}\sigma(\omega).$$

Once we know the position of the initial vertex, we perform a lookup in the importance *kd*-tree to find the region that this vertex belongs to and the environment map with approximate incident importance. Using this additional information, it is possible to sample the outgoing direction from the light source using the mixture density,

$$p(\boldsymbol{\omega}) = c_1\, p_{L_\mathrm{e}}(\boldsymbol{\omega}) + c_2\, p_{W_\mathrm{i}}(\boldsymbol{\omega}), \tag{5.17}$$

where $p_{L_\mathrm{e}}(\boldsymbol{\omega})$ is a PDF that is proportional to $L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_0, \boldsymbol{\omega})\,|\cos\theta|$. The constants, c_1 and c_2, control the influence of the radiance term versus the importance term. The effectiveness of the algorithm is not particularly sensitive to these values, so in our experiments we have simply used $c_1 = c_2 = 0.5$.

Ideally we would like to have sampled the outgoing direction according to

$$p(\boldsymbol{\omega}) \propto L_{\mathrm{e},\partial\mathcal{V}}(\mathbf{x}_0, \boldsymbol{\omega})\, W_\mathrm{i}(\mathbf{x}_0, \boldsymbol{\omega})\,|\cos\theta|,$$

i.e., according to product distribution of radiance and importance. Compared to this, it is clear that Equation 5.17 is only an approximation. See Section 5.4.3 for further discussion of this issue.

5.3.2 Camera Sampling

Sampling the initial vertex in the eye path turns out to be simpler than sampling the first vertex in the light path. As discussed in Section 5.2.1, we simply sample the initial vertex with uniform probability across the aperture.

It is possible to sample the outgoing direction using a procedure similar to how the outgoing direction from light sources is found. I.e., find the incident radiance by querying the radiance *kd*-tree and then sample a direction using a mixture density based on PDFs proportional to W_e and L_i. However, this is often not a good idea. The reason is that while the this density may in fact reduce absolute error, it can result in a very uneven distribution of samples among the different measurements (the W_e^j's), since the number of samples allocated to a given measurement will depend on how bright the corresponding pixel is. As a result, dimmer pixels will be estimated with a relatively lower accuracy than brighter pixels, which will manifest itself as noise in the shadows, something to which the human visual system is particularly susceptible. Instead, we sample the outgoing direction according to W_e as defined by the camera model.

5.3.3 Kernel Sampling

We also modify the procedure for sampling outgoing directions at the remaining vertices in the random walks. If $\mathbf{x}_n \in \partial\mathcal{V}$, the usual approach is to sample the outgoing direction according to a pair of PDFs, p_{f_s} and $p_{f_\mathrm{s}^*}$. The first PDF, p_{f_s}, is proportional to $f_\mathrm{s}(\mathbf{x}_n, \boldsymbol{\omega}, \boldsymbol{\omega}')\,|\cos\theta|$ and is used when constructing eye paths and the second PDF, $p_{f_\mathrm{s}^*}$, is proportional to $f_\mathrm{s}^*(\mathbf{x}_n, \boldsymbol{\omega}, \boldsymbol{\omega}')\,|\cos\theta|$ and is used when constructing light paths.

We augment these densities by a term that accounts for incident radiance or importance. Like above, we query the *kd*-trees to find the current region and the incident quantities. This makes it possible to use the mixture density,

$$p(\boldsymbol{\omega}) = c_1\, p_{f_\mathrm{s}}(\boldsymbol{\omega}) + c_2\, p_{L_\mathrm{i}}(\boldsymbol{\omega}), \tag{5.18}$$

for constructing eye paths and the mixture density,

$$p(\boldsymbol{\omega}) = c_1\, p_{f_\mathrm{s}^*}(\boldsymbol{\omega}) + c_2\, p_{W_\mathrm{i}}(\boldsymbol{\omega}), \tag{5.19}$$

for constructing light paths. Note that it is necessary to query both radiance and importance even if only one of these quantities is ever used for constructing a particular path. The reason is that this information is required for computing the probabilities of the other ways the path could have been constructed, which is necessary for doing multiple importance sampling.

The functions, p_{f_s} and $p_{f_\mathrm{s}^*}$, may themselves be mixture densities. This is common if the BSDF has multiple components. For instance, plastic can modeled with a BSDF for the base layer (e.g. a Lambertian BSDF) combined with a BSDF for the coating (e.g. a Cook-Torrance BSDF). In that case, p_{f_s} (and $p_{f_\mathrm{s}^*}$) would be mixture densities with a term for each BSDF component and the final mixture density would then have three components. We assign equal weights to all the densities, so that $c_1 = c_2 = \ldots = c_n = 1/n$, if there are n components in the mixture.

If the BSDF (and thereby p_{f_s}) contains a Dirac delta function, the above procedure cannot be used. The reason is that for these materials (perfect mirrors, glass, etc.) only single incoming directions matter, and attempting to find these by random sampling according to incident radiance or importance is not possible. Instead, we simply sample these directions according to p_{f_s}. However, incident radiance or importance can still be used for selecting among specular BSDF components if the BSDF has more than one. For instance, glass is often modeled using the BSDF for perfect specular reflection combined with the BSDF for perfect specular transmission (see Section 2.6.1). The standard way of selecting between these two components is to use the Fresnel coefficients. However,

if we know the incident quantities, we can look up the approximate incoming radiance or importance from the reflected and refracted directions using the environment maps. We can then base our decision on these values scaled by the Fresnel coefficients.

Sampling outgoing directions for the volume case ($\mathbf{x}_n \in \mathcal{V}_0$) is very similar to the above procedure. We first query the *kd*-tree so that incident radiance and importance can be estimated. Then rather than sampling the outgoing direction according a PDF, $p_{f_\mathrm{p}}(\boldsymbol{\omega}) \propto f_\mathrm{p}(\boldsymbol{\omega} \cdot \boldsymbol{\omega}')$, we use the mixture density,

$$p(\boldsymbol{\omega}) = c_1\, p_{f_\mathrm{p}}(\boldsymbol{\omega}) + c_2\, p_{L_\mathrm{i}}(\boldsymbol{\omega}),$$

for constructing eye paths and the mixture density,

$$p(\boldsymbol{\omega}) = c_1\, p_{f_\mathrm{p}}(\boldsymbol{\omega}) + c_2\, p_{W_\mathrm{i}}(\boldsymbol{\omega}),$$

for the light paths. Like above, we assign equal weights to all the components in the mixture.

5.3.4 Russian Roulette

Global context can also be used to improve Russian roulette, so that paths that are heading toward unimportant regions of the scene are terminated with higher probability. In order to implement this we first sample a candidate outgoing direction, $\boldsymbol{\omega}'$, according to the $p(\boldsymbol{\omega}')$ given above. We then perform a look up in the appropriate environment map to find the incident importance or radiance from that direction.

For light paths the survival probability can be computed as

$$q = \min\left\{1, \frac{\alpha\, f_\mathrm{s}^*(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}', \boldsymbol{\omega})\, W_\mathrm{i}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}')\, |\cos\theta|}{c_1\, p_{f_\mathrm{s}^*}(\boldsymbol{\omega}) + c_2\, p_{W_\mathrm{i}}(\boldsymbol{\omega})}\right\},$$

and for eye paths it can be computed as

$$q = \min\left\{1, \frac{\beta\, f_\mathrm{s}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}', \boldsymbol{\omega})\, L_\mathrm{i}(\mathbf{x}_\mathrm{s}, \boldsymbol{\omega}')\, |\cos\theta|}{c_1\, p_{f_\mathrm{s}}(\boldsymbol{\omega}) + c_2\, p_{L_\mathrm{i}}(\boldsymbol{\omega})}\right\}.$$

The constants, α and β, are necessary since neither W_i or L_i are normalized. The procedure for computing the survival probability in the volume case is almost identical, but of course uses the phase function instead.

5.4 Results

We have implemented a spectral version of the Metropolis light transport algorithm in C/C++ (using the variation by Kelemen et al. [2002]) that we will use to demonstrate the enlightened local path sampling algorithm.

5.4.1 Sampling using Adjoint Measurements

We begin by examining the advantages of sampling starting positions on light sources using adjoint measurements. The algorithms used are bidirectional path tracing and Metropolis light transport modified to sample starting positions in light paths according to the adjoint image, rather than according to radiant exitance alone.

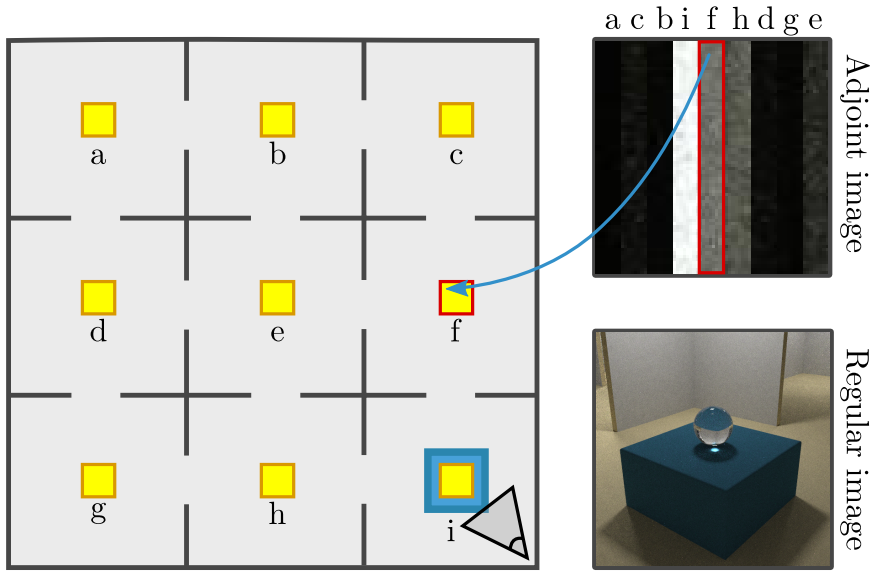
The test scene, shown in Figure 5.10, is a maze-like scene of 3×3 connected rooms. Each room has a diffuse area light source labeled a–i and all the light sources have the same power. The camera is located in the bottom right room observing a glass sphere on a blue slab. Light source 'i' is responsible for the majority of lighting is this room, though the camera is also able to see a small part of the adjacent rooms, which are lit primarily by light source 'f' and 'h'.

The right part of Figure 5.10 shows the adjoint image for this scene, which was rendered in the resolution of 72×72 using regular Metropolis light transport. Due to the way positions in $[0; 1]^2$ are mapped to the light sources, the appearance of the adjoint image is that of nine almost solid colored columns (some Monte Carlo noise is still visible). These nine 8×72 sub-images corresponds to the adjoint measurements across the light sources. E.g. light source 'f', which is highlighted in red, corresponds to the fifth of these sub-images.

The reason the nine columns are almost solid colored is that the incident importance across the light sources in almost constant (the light sources are small planar quadrilaterals). Since the radiance is also constant, the adjoint measurements become near constant across the light sources (recall the definition of adjoint measurements, Equation 5.6). This is only true for simply scenes such as this; for scenes with more complicated area light sources the adjoint measurements can vary considerably across a light source.

While each sub-image is of relatively constant intensity, the difference in intensity between the columns is considerable. Not surprisingly, light source 'i' in the room containing the camera is the most important by far, so we should expect that spending most time on this light source would be beneficial. The

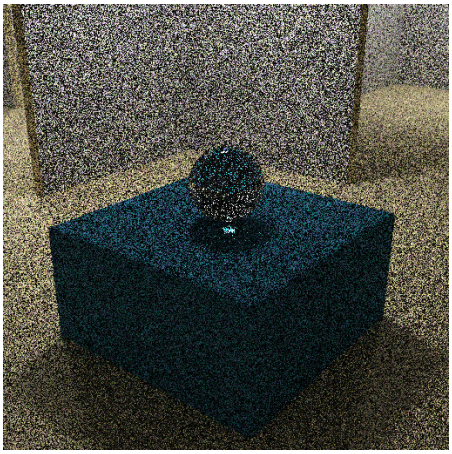
Figure 5.10: Test scene used to demonstrate sampling using adjoint measurements. This scene consists of 3×3 connected rooms each with an area light source in the ceiling. All the nine light sources have the same power and are labeled a–i. The camera is in the room in the bottom right observing a glass sphere on a blue slab. The adjoint image of the scene is shown in the right part of the figure together with the regular image seen from the camera. Each column in the adjoint image corresponds to adjoint measurements of one of the nine light sources.



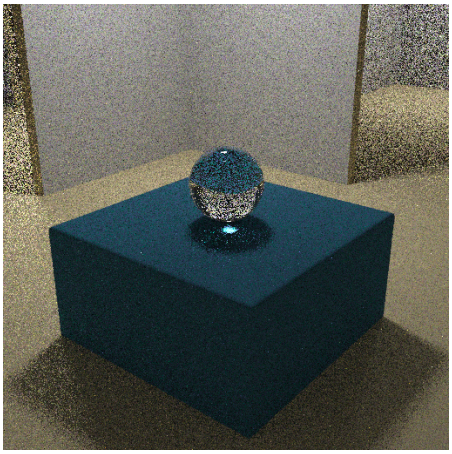
light sources in the rooms directly adjacent to the room containing the camera are also relatively important ('f' and 'h') and so is light source 'e'. However, the remaining light sources are all relatively unimportant, since they contribute very little light visible to the camera, and should therefore only be assigned few computational resources.

Figure 5.11 shows the results of rendering the maze scene using bidirectional path tracing and Metropolis light transport. The top row of images were rendered using bidirectional path tracing and the bottom row using Metropolis light transport. The left column of images uses regular sampling of light path starting positions, whereas the right column uses the adjoint image seen in Figure 5.10. All images were rendered in approximately the same time (the overhead introduced by warping samples according to the adjoint image is negligible compared to the cost of finding intersections, sampling BSDFs etc.). The images rendered

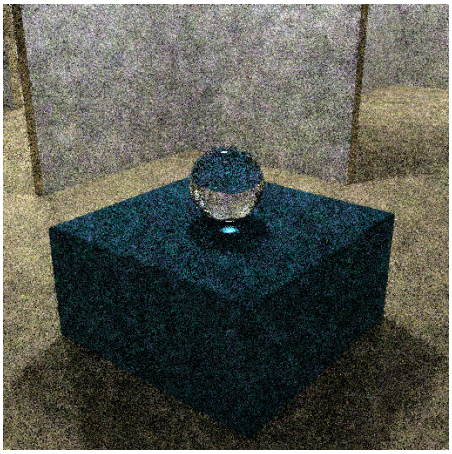
Figure 5.11: Results of rendering the maze scene with and without the proposed technique using bidirectional path tracing and Metropolis light transport. All the images were rendered using four samples per pixel, which took approximately one minute.



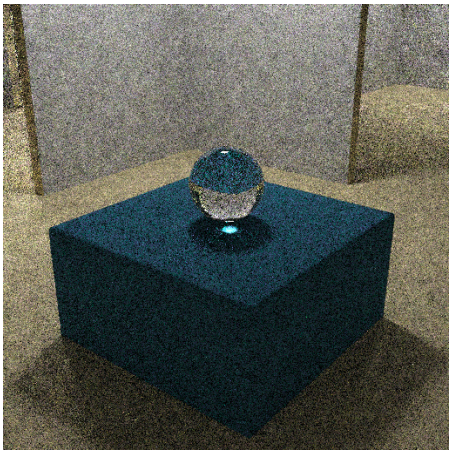
(a) Bidirectional path tracing (using radiant exitance).



(b) Bidirectional path tracing (using adjoint measurements).



(c) Metropolis light transport (using radiant exitance).



(d) Metropolis light transport (using adjoint measurements).

using bidirectional path tracing used four samples per pixel. The images rendered using Metropolis light transport used four samples per pixels on average, since due to the nature of this algorithm it is not possible to stratify the samples exactly. The rendering time does not include the time required to generate

the adjoint image. However, the low resolution of the adjoint images (72×72), compared to the resolution of the final image (512×512) means that the cost of generating the adjoint image is very small (e.g. the cost of rendering the adjoint image at 50 samples per pixel is approximately the same as rendering the full image at one sample per pixel).

The benefit of sampling according to the adjoint image when using bidirectional path tracing can be seen by comparing Figure 5.11a and Figure 5.11b. With regular bidirectional path tracing many light paths are started on the unimportant light sources. Since it is unlikely that these random walks will end up near the eye path, most of the shadow feelers will be blocked and consequently many samples will be zero. On the other hand, if the adjoint image is used, most light paths will start on the light source in the room with the camera or in the rooms directly adjacent to this. These paths will have a much higher probability of successfully connecting to the eye path and therefore have lower variance.

As shown in Figure 5.11, the Metropolis light transport algorithm also benefits from sampling according to the adjoint image, though to a lesser extent than bidirectional path tracing. If the noise in Figure 5.11c and Figure 5.11d is compared it can be seen that the noise in the left image is more splotchy, whereas the amount of noise in the right image is less and in addition the noise has a more "pleasant" uniform distribution. The reason for this difference is related to the discussion of bidirectional path tracing above, since the Metropolis light transport algorithm uses this algorithm to generate paths.

Recall that Metropolis light transport is based on the idea of mutating paths using two mutation strategies (see Section 4.7.4), Also recall that a good mutation is a mutation that significantly changes the current state, while still being accepted with high probability, since such mutations quickly explore the state space. The first mutation strategy is responsible for small mutations and the second is responsible for large mutations (these are implemented using the random walk kernel and the independence kernel described in Section 3.8.2). We chose each type with equal probability. The reason for the appearance of Figure 5.11c is that most large mutations are rejected, since the proposed paths do not connect the sensor and light sources, and thus transfer no energy. This means that many samples will be repeated and the chain will move only because of small mutations. This leads to highly correlated samples and the splotchy patterns in the noise. This is not the case in Figure 5.11d, since here the adjoint image is used to find starting points for light paths. Large mutations are therefore more likely to be accepted and the state space is explored more quickly.

5.4.2 Metropolis Particle Sampling

The algorithms of bidirectional particle sampling and Metropolis particle sampling were presented in Section 5.2.2. The advantage of these algorithms compared to classical particle sampling is that the distribution of particles is based on both radiance and importance, rather than only one of these quantities. Since Metropolis particle sampling is really an extension of bidirectional particle sampling, we will focus on this method here.

In order to demonstrate Metropolis particle sampling, we will use the scene shown in Figure 5.12. This scene, which is similar to the test scene in Veach and Guibas [1997], consists of two rooms. The camera is observing a table with some objects in one of the rooms. The only light source in the scene is an area light source in the ceiling of the other room. The door between the rooms is almost closed allowing only small fraction of the emitted light to reach the room with the camera.

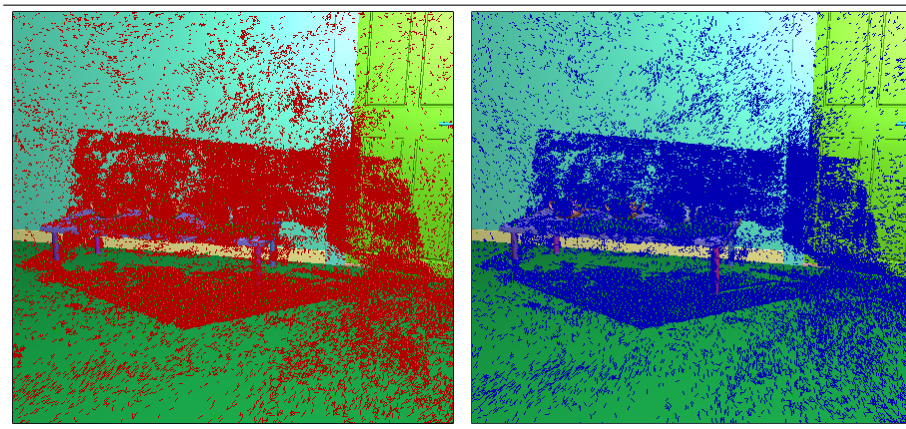
Figure 5.12: Test scene used to demonstrate the proposed technique. This scene, which is inspired by Veach and Guibas [1997], consists of two rooms separated by a door slightly ajar. The scene has one light source, which is an area light source in the ceiling of the adjoining room. This scene is lit almost exclusively by indirect illumination.

This scene is difficult, because the distribution of radiance is very different from

the distribution of importance. This means that if photons and importons are sampled using classical particle sampling, very few photons would end up near the sensors and very few importons near the light source.

As shown in Figure 5.13, this is not the case with Metropolis particle sampling. Since this algorithm samples particles using paths starting from both sensors and light sources, many photons end up in visible parts of the scene. Similarly, the density of importons in the adjoining room is much higher with Metropolis particle sampling than it would have been with classical particle sampling.

Figure 5.13: Photons (left) and importons (right) generated using Metropolis particle sampling. Note the similarity between the distributions of photons and importons. Also note the much high particle density inside the camera frustum (the camera location is that of Figure 5.12). Had the photons been sampled using classical photon sampling, the vast majority of photons would have been located in the adjoining room and only very few would have found their way to the camera.



(a) Distribution of photons. **(b)** Distribution of importons.

Another thing to notice is that the distribution of photons and importons is very similar, as they should be according to Equation 5.14. This is due to the way the particles are generated using Metropolis sampling and bidirectional particle sampling, since with these algorithms a photon and an importon is extracted from each vertex along the generated paths. One consequence of this is that the density of particles in visible areas will be high, which is seen in the image as the outline of the camera frustum.

5.4.3 Sampling Directions using Incident Quantities

Previously we saw how the adjoint image could be used to improve sampling of light path start positions by sampling according to the product of emitted radiance and incident importance. As discussed in Section 5.3.3, this idea can be extended to sampling outgoing directions at the remaining vertices in the path, so that eye paths are sampled according to both the scattering kernel and incident radiance and light paths according to the adjoint scattering kernel and incident importance. Unfortunately, as discussed further below, sampling outgoing directions at each vertex using these incident quantities turns out to be more challenging than sampling starting positions in light paths.

To test the proposed algorithm we will use the scene shown in Figure 5.12. This scene is interesting, since it is very difficult to generate paths that randomly pass through the small opening in the doorway using regular bidirectional path tracing. We first built the required data structures as described in Section 5.2. This involved generating particles using Metropolis particle sampling. The photons were then organized into approximately 16384 clusters (shown in Figure 5.9) and the same was done for the importons. Finally, the environment maps with incident radiance or importance were computed (we used a resolution of 32×16). See Table 5.1 for details. For this particular scene there is no advantage to sampling light path start positions using the adjoint image, since the scene only has one light source, which is a simple planar area light source, so that part of the algorithm has been elided.

Table 5.1: Various statistics for the preprocessing pass that was used for generating the picture shown in Figure 5.14a. The total time for the preprocessing pass was 121 seconds. Particles is the total number of photons and importons that were sampled. The memory usage includes both the *kd*-tree and environment maps.

	Particles traced	Clusters	Resolution	Memory usage
Radiance	85×10^6	16384	32×16	40 MiB
Importance	85×10^6	16384	32×16	40 MiB

As described in Section 5.3, the basic idea of enlightened local path sampling is to guide light paths to areas visible to the camera and eye paths to areas "visible" to the light sources. In the present scene this means that the outgoing direction from the initial vertex in the light path should be chosen in the general direction of the doorway (based on Equation 5.17). The same is true for the remaining vertices in the light path (but instead using Equation 5.19), and also for vertices in the eye path (using Equation 5.18). Ideally, this should lead to a higher probability of successfully generating a path that connects the light

source and the sensor.

Figure 5.14: Results of rendering using the proposed algorithm. The top image was rendered in five minutes, which does not include the 121 seconds for the preprocessing pass. The bottom image was rendered using regular Metropolis light transport in seven minutes.

(a) Metropolis light transport (using enlightened local path sampling).

(b) Metropolis light transport (using regular local path sampling).

Figure 5.14 shows the results of rendering with and without the proposed technique on an Intel Core 2 Quad (Q6600) using four threads. Both images were rendered in approximately seven minutes. Compared to the technique of sampling starting positions using the adjoint image, it is clear that the benefit of sampling directions in this way is much more questionable (in fact in this particular scene it even seems to be detrimental to the overall image quality). As with any importance sampling scheme, the goal is to increase to efficiency. Since the preprocessing pass takes time from the rendering pass and each sample in addition is more expensive using the proposed method, the total number of samples is less compared to regular Metropolis light transport. This means that for the proposed method to be of any benefit, the decrease in variance of the samples must more than make up for this, since otherwise efficiency is reduced. Unfortunately this does not seem to be the case, since Figure 5.14a has more variance than Figure 5.14b.

There are several possible causes of the negative results. Firstly, the environment maps with incident radiance/importance had not converged completely, so some Monte Carlo noise was still present. This could have negatively impacted the sampling. Secondly, the resolution of the environment maps may not have been sufficient. In the test scene the small slither in the doorway is a very small target, which could not be accurately represented in a 32×16 image. Thirdly, the clusters may have been too large. This would mean that the representation of the incident quantity would have been inaccurate with negative consequences for sampling. Finally, the overhead of sampling according to additional PDFs also negatively affects the results. These are causes that could be fixed by assigning more resources (both time and memory) to the preprocessing pass, though at the cost of further reducing the resources available for the rendering pass. Alternative options for addressing these issues are discussed in Section 5.6.

5.5 Related Work

In the following we discuss existing algorithms and point out their differences and similarities to the present work.

5.5.1 Particle Sampling

As described earlier, particle sampling is used to create the data structures we use to improve importance sampling. Particle sampling has been used extensively in computer graphics. The most popular algorithm based on particle

sampling is the photon mapping algorithm by Jensen [1996], which was later extended to participating media by Jensen and Christensen [1998]. Photon mapping is a two-pass algorithm, where photons are generated in the first pass using the procedure we called classical particle sampling earlier. In the second pass the scene is rendered from the view point and radiance estimated using density estimation based on the photons. Since this algorithm uses classical particle sampling, it suffers from problem of unsuitable particle distributions described earlier, i.e. only few photons may reach areas visible to the camera. Peter and Pietrek [1998] propose a three-pass algorithm to remedy this situation. Their idea is to first trace importons through the scene and then use these to guide photons to visible areas. They also propose a way of sampling the starting positions on the light sources based on importance, which is similar in spirit to how we select starting positions on the light sources using adjoint measurements. However, while using this algorithm could potentially result in better photon distributions in some scenes, it is still based on classical particle sampling. Therefore, in difficult scenes too few importons will reach areas near the light sources where they would be needed later in the second pass to guide the photons. Simpler variations of this approach are described by Suykens and Willems [2000] and by Keller and Wald [2000]. They both trace photons using classical particle sampling, but then randomly decide whether or not to store the photon based on the importance. This can help reduce the number of photons in unimportant regions, but does not help to increase the photon density in important regions.

Our particle sampling approach is closer to that of Fan, Chenney, and Lai [2005], who also sample particles from paths using Metropolis-Hastings based on the measurement contribution function. This leads to a distribution of photons that is based on both radiance and importance. However, their motivation is different from ours. Their goal is to generate a distribution of photons that is well suited for final gathering in a traditional photon mapping context. In contrast, we sample both photons and importons and use these particles to improve the sampling of paths in bidirectional path tracing. An additional difference is that their algorithm is based on the original formulation of Metropolis light transport, whereas we used the Kelemen variation.

5.5.2 Radiance/Importance Driven Path Sampling

Many algorithms used radiance or importance to improve path sampling. An early example of radiance driven path sampling is the two-pass algorithm by Jensen [1995]. In the first pass a photon map is constructed similar to regular photon mapping and in the second pass the scene is rendered using path tracing. During path tracing, the outgoing direction at each vertex is sampled using a

PDF based on the product of the BRDF, cosine term, and the incident radiance, which is estimated using the photon map. A variation of this approach described by Hey and Purgathofer [2002], which differs only in how photons are used. Lafortune and Willems [1995b] also use radiance driven path tracing, though in their case radiance is stored in a 5D "binary" tree that is constructed during rendering.

An algorithm for importance driven path sampling is described by Dutré and Willems [1994]. This algorithm is a variation of light tracing, where adaptive PDFs are constructed at the (point) light sources, so that outgoing directions of high importance can be chosen with higher probability. This algorithm was extended by Dutré and Willems [1995] to also take importance into account when sampling outgoing directions at the remaining vertices. This is achieved by discretizing the scene into a number of patches that each store incident importance, which is quite similar to how we divide the scene into regions.

The biggest difference, compared to the above algorithms, is that our algorithm is based on bidirectional path tracing. In most cases, this algorithm is much more efficient than path tracing or light tracing, especially if combined with Metropolis light transport. Since this algorithms is bidirectional, we generate light paths using importance and eye paths using radiance. This means that our algorithm can be seen as a combination of radiance driven path tracing and importance driven light tracing, but combined based on the weights given by multiple importance sampling.

5.5.3 (Ir)radiance Caching

Indirect illumination changes smoothly across surfaces and is therefore well suited for interpolation. This is exploited in the irradiance caching algorithm by Ward, Rubinstein, and Clear [1988], which stores (indirect) irradiance in an octree. During rendering the octree is queried for irradiance samples near the current vertex. If suitable samples are found the irradiance is computed by interpolation; otherwise the irradiance is computed using Monte Carlo integration and the octree updated with the new irradiance value. This algorithm was later improved by Ward and Heckbert [1992] with gradients for higher quality interpolation.

Since the stored quantity is irradiance, only Lambertian (ideal diffuse) can be handled with irradiance caching. Radiance caching, developed by Křivánek, Gautron, Pattanaik, and Bouatouch [2005b] and improved with gradients by Křivánek, Gautron, Bouatouch, and Pattanaik [2005a], caches radiance encoded as low-order hemispherical harmonics. This makes it possible to also handle

low-frequency BRDFs. Radiance caching has also been extended to participating media by Jarosz, Donner, Zwicker, and Jensen [2008a] and improved with gradients by Jarosz, Zwicker, and Jensen [2008b].

Unlike the algorithm presented in this work, these algorithms are all biased, since they use interpolated values. However, there are still many similarities, especially with the radiance caching algorithms. For instance, in our algorithm it would also have been possible to store incident radiance/importance as low-order spherical harmonics. This could potentially have reduced memory requirements and would have made it possible to use gradients for better interpolation. Functions encoded as spherical harmonics can also be importance sampled directly (using the method presented in Jarosz, Carr, and Jensen [2009]). However, we found that compared to our tabulated CDF representation, this way of importance sampling is significantly slower, since we can only warp a single sample at a time.

5.5.4 Exploiting Coherence

Several other algorithms gain efficiency by exploiting various forms of coherence. Clarberg and Akenine-Möller [2008] encode the 4D visibility function sparsely as 2D visibility maps stored in a *kd*-tree. During rendering the nearest sample with similar normal is located and used to improve the efficiency of the Monte Carlo integration using the technique of control variates. This is very similar to our approach, except that we store radiance/importance rather than the binary visibility function.

Budge, Anderson, and Joy [2008] present an algorithm for rendering unbiased caustics lighting in a virtual point light source algorithm. Like ours, this algorithm is based on tracing particles (in this case caustics photons) and clustering them. In each cluster the incident radiance from caustics is computed and represented in tabulated form. This in turn makes it possible to importance sample using the inversion method. Our algorithm is essentially a generalization of this algorithm to any kind of lighting effect (i.e. not just caustics). Unlike their algorithm, we also need to represent importance, since we use bidirectional path tracing as the underlying algorithm rather than virtual point light sources.

Coherence also exists between pixels in an image, a fact which is exploited in the algorithm by Cline, Adams, and Egbert [2008]. This algorithm works by constructing "importance maps," which are similar to our tabulated CDFs. Like our algorithm, these maps are then used to warp the random number used for sampling paths. However, unlike our algorithm, these maps are constructed "lazily" based on previously rendered pixels, rather than in a preprocess.

5.6 Future Work

In the following we discuss possible improvements to the presented algorithm that could be interesting to pursue as future work.

5.6.1 Hierarchical Basis Functions

Hierarchical basis functions, such as wavelets or spherical harmonics, could potentially be used to improve the procedure for sampling outgoing directions at vertices. Recall that rather than sampling according to the product of the scattering kernel and incident radiance/importance, we sample according either factor separately and then combine the results using multiple importance sampling (see Section 5.3.3). Multiple importance sampling works well if the functions are relatively similar (i.e. if they "overlap"). However, if the functions are very dissimilar, multiple importance sampling can be inefficient, since few samples will be placed where the product of the functions is large.

Clarberg, Jarosz, Akenine-Möller, and Jensen [2005] propose a method for sampling according to the product distribution of functions represented using Haar wavelets. This work is generalized to spherical harmonics by Jarosz et al. [2009]. Using these methods would make it possible to sidestep the problems with multiple importance sampling mentioned above. The difficulty with using these methods is that the scattering kernel would have to be available in the chosen basis. The is not a problem for measured BRDFs, since they can be stored directly as e.g. wavelets. However, obtaining the wavelet coefficients of a 2D slice of an analytical reflectance model or phase function is more challenging. Another issue is the time it takes to generate a single sample. These methods were presented in the context of computing direct lighting from environment maps, where many samples can be warped simultaneously, which makes the per sample overhead of the warping algorithm small. However, as mentioned earlier, this is not possible when using the method for creating random walks, since in that case only a single outgoing direction is needed at each vertex. This makes this form of sampling fairly costly compared to traditional methods, such as BSDF importance sampling for analytical reflection models.

5.6.2 Gradient Based Interpolation

As mentioned, storing incident radiance and importance as spherical harmonics rather than as tabulated functions may decrease storage requirements and even

enable sampling according to the full product distribution.

Another advantage of representing these quantities as spherical harmonics is that it could make the approximations of the functions more accurate. Recall that radiance and importance is currently approximated as being constant with respect to position within each region. If a spherical harmonic approximation is used instead it becomes possible to reconstruct the quantities by interpolating the coefficients in the nearest regions. I.e., rather than simply using the coefficients stored in the nearest region, we find the n nearest regions and interpolate the coefficients stored in them. If this approach is used in combination with the gradients derived by Křivánek et al. [2005a] and by Jarosz et al. [2008b] the resulting approximation should be significantly improved.

5.6.3 Progressive Construction of Global Context

Two-pass algorithms have the disadvantage that it may be difficult to determine in advance how many computational resources to assign to each pass. In the present case this means that it may be difficult to determine how much time to allocate to constructing global context versus the actual rendering process.

This problem can be avoided if the algorithm is transformed into a single pass algorithm, where the approximations of radiance and importance in the scene are constructed progressively during rendering. This should be possible, since the preprocessing pass and rendering pass are quite similar in that they both sample paths. The difference is just that in the preprocessing pass the paths are used to build global context, whereas in the rendering pass they are used to create the image. During rendering the paths would be used both to create the image and to update the global context and associated PDFs. Once updated rendering would then continue with the new improved PDFs.

The difficulty with this approach is that changing PDFs during rendering when using regular Monte Carlo can be tricky. In general it will be necessary to discard all samples generated using the previous PDFs to ensure unbiasedness, which of course is wasteful. Alternatively the samples can be reweighted, though this would require storing all previously generated samples, which is unpractical. A solution to this problem may be offered by Population Monte Carlo methods. Population Monte Carlo [Cappé, Guillin, Marin, Robert, and Roberty, 2004] is an alternative formulation of Monte Carlo based on evolving a population of samples through successive generations. The advantage of approach is that the algorithm is unbiased even if the PDFs are updated after each generation.

Chapter 6

Conclusion

The light transport problem can be phrased as an integration problem that can be solved using Monte Carlo methods. Unfortunately many interesting scenes are difficult to render using these methods, due to lack of good methods for importance sampling. The fundamental difficulty that limits effective importance sampling is lack of global context; i.e. we do not have enough information available to sample paths distributed according to the measurement contribution function, which would be ideal according to the principle of importance sampling. Consequently, generating images using these methods can in some cases be extremely time consuming.

The goal of this work was to investigate how additional information derived from the scene description could be used to improve importance sampling. Our focus was on modern unbiased methods, in particular bidirectional path tracing and Metropolis light transport.

The challenge of any importance sampling strategy is that the additional time spent must be more than offset by the reduction in variance the method provides, since otherwise overall efficiency is reduced. In our case this meant that the relative decrease in variance of the samples had to be greater than the relative increase in rendering time caused by having to analyze the scene, extract the relevant information, and build data structures.

We have investigated two methods for improving importance sampling that we together call enlightened local path sampling and that uses this kind of derived information. The first method is simple to implement and provides good speedups in scenes with many or complicated light sources. The second method is more complicated and unfortunately did not provide the expected speedup.

We have also derived a new method of particle sampling. This was necessary, since existing methods were unsuitable for our purposes.

6.1 New Ways of Sampling Particles

Particle sampling forms the basis of many algorithms in computer graphics. The standard way of generating photons is to perform a random walk starting from the light source. The downside to this approach (and the similar approach to create importons) is that the particles become distributed according to radiance (or importance). This is undesirably in scenes where the distribution of radiance and importance differ greatly, which is common in hard scenes.

To avoid this problem we suggested bidirectional particle sampling. Bidirectional particle sampling samples photons and importons simultaneously from paths built starting from both sensor and light source. Metropolis particle sampling extends this method by sampling paths according to the measurement contribution function so that particles are sampled where radiance and importance are simultaneously high.

The advantage of this approach is that regions where many important paths pass through are sampled more densely. This means that more samples are placed where they matter most to the integral being solved.

6.2 Sampling using Adjoint Measurements

Sampling the initial vertex in light paths in bidirectional path tracing can be difficult in scenes with many light sources. The problem is that based on the scene description we only know the power of the light sources, but not how important the emitted light is to the sensors. To help solve this problem we introduced the concept of adjoint measurements. Adjoint measurements are the light sources' response to incident importance. This makes adjoint measure-

ments ideal for importance sampling light path start positions. Computing the adjoint measurements can be done in a preprocess and is fairly fast.

Implementing this method in a existing bidirectional path tracer is fairly straightforward and only requires a modest amount of coding. The advantage of using this method can be large in scenes with many light sources and we are able to demonstrate significant reduction in variance both for bidirectional path tracing and Metropolis light transport.

6.3 Sampling using Incident Quantities

Bidirectional path tracing can be inefficient in scenes where the distribution of radiance and importance differ greatly. The reason is that light paths and eye paths explore different parts of the scene and therefore never "meet." This happens because the random walks are based on either radiance or importance, but not both.

To solve this problem we suggested also taking incident importance into account when sampling vertices in the light paths and incident radiance into account when sampling vertices in the eye paths. This is similar in spirit to sampling using adjoint measurements and the effect would be to make light paths more inclined to move to areas of high importance and eye paths more inclined to visit brightly lit areas.

The difficulty with this approach is that in order to use it we need a representation of radiance and importance in the scene. Storing and computing these quantities accurately is much more difficult than calculating and storing the 2D adjoint image due to the higher dimensional nature of these functions.

To keep memory requirements manageable we used a compact data structure that stored the quantities sparsely as low resolution environment maps. As discussed in the previous chapter this was not ideal for a number of reasons, which lead to disappointing results. However, as outlined in the section on future work, other ways of representing these functions exist and may be worth exploring.

We also discussed how this technique could be extended to the volume case. This was achieved by storing a representation of radiance/importance inside the cells of the scene, in addition to storing the quantities across the boundaries of the cells. The idea was that this could potentially allow for more effective importance sampling in the presence of participating media and subsurface

scattering. However, due to time constraints we did not implement this. Consequently, more work is necessary to determine whether this approach is really beneficial.

Bibliography

Arthur Appel. Some techniques for shading machine renderings of solids. In *AFIPS '68 (Spring): Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45, New York, NY, USA, 1968. ACM. [7]

James Arvo. Backward Ray Tracing. In *In ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, pages 259–263, 1986. [8]

James Arvo. Transfer Equations in Global Illumination. In *Global Illumination, SIGGRAPH '93 Course Notes*, August 1993. [16, 88]

James Arvo. *Analytic Methods for Simulated Light Transport*. PhD thesis, Yale University, December 1995a. [90]

James Arvo. The Role of Functional Analysis in Global Illumination. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 115–126. Springer-Verlag, New York, 1995b. [90]

James Arvo and David Kirk. Particle Transport and Image Synthesis. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 63–66, New York, NY, USA, 1990. ACM. ISBN 0-89791-344-2. [104]

Michael Ashikhmin, Peter Shirley, and Brian Smits. A Variance Analysis of the Metropolis Light Transport Algorithm. *Computers and Graphics*, 25:287–294, 2001. [115]

Ronen Barzel. Lighting Controls for Computer Cinematography. *J. Graph. Tools*, 2(1):1–20, 1997. ISSN 1086-7651. [49]

Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975. [146]

Blender Community. Phong interpolation and ray trace shadows, August 2004. `http://www.blender.org/development/release-logs/blender-234`. [43, 44]

James F. Blinn. Simulation of Wrinkled Surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 286–292, New York, NY, USA, 1978. ACM. [40]

Brian C. Budge, John C. Anderson, and Ken Joy. Caustic Forecasting: Unbiased Estimation of Caustic Lighting for Global Illumination. In *Proc. of Pacific Graphics*, October 2008. [164]

Olivier Cappé, Arnaud Guillin, Jean-Michel Marin, Christian P. Robert, and Christian P. Roberty. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13:907–929, 2004. [166]

Nathan A. Carr, Jesse D. Hall, and John C. Hart. The Ray Engine. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 37–46, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. ISBN 1-58113-580-7. [9]

S. Chandrasekhar. *Radiative Transfer*. Dover Publications, Inc., 1960. [16]

Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335, 1995. [81]

Per H. Christensen. Adjoints and Importance in Rendering: An Overview. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):329–340, 2003. ISSN 1077-2626. [93]

Petrik Clarberg and Tomas Akenine-Möller. Exploiting Visibility Correlation in Direct Illumination. *Computer Graphics Forum*, 27(4):1125–1136, 2008. [164]

Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. In *Proceedings of ACM SIGGRAPH 2005*. ACM Press, July 2005. [165]

David Cline, Justin Talbot, and Parris Egbert. Energy Redistribution Path Tracing. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1186–1195, New York, NY, USA, 2005. ACM. [116]

David Cline, Daniel Adams, and Parris K. Egbert. Table-driven Adaptive Importance Sampling. *Computer Graphics Forum*, 27(4):1115–1123, 2008. [164]

M. Cohen and J. Wallace. *Radiosity and Realistic Image Synthesis*. Addison-Wesley, 1993. [16]

Michael Cohen, Donald Greenberg, David Immel, and Philip Brock. An Efficient Radiosity Approach for Realistic Image Synthesis. *IEEE Comput. Graph. Appl.*, 6(3):26–35, 1986. ISSN 0272-1716. [7]

Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A Progressive Refinement Approach to Fast Radiosity Image Generation. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 75–84, New York, NY, USA, 1988. ACM. ISBN 0-89791-275-6. [7]

R. L. Cook and K. E. Torrance. A Reflectance Model for Computer Graphics. *ACM Trans. Graph.*, 1(1):7–24, 1982. ISSN 0730-0301. [106]

Robert L. Cook. Stochastic Sampling in Computer Graphics. *ACM Trans. Graph.*, 5(1):51–72, 1986. ISSN 0730-0301. [52]

Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145, New York, NY, USA, 1984. ACM. ISBN 0-89791-138-5. [8, 107]

Paul Debevec. Rendering with Natural Light. In *SIGGRAPH '98: ACM SIGGRAPH 98 Electronic art and animation catalog*, page 166, New York, NY, USA, 1998. ACM. ISBN 1-58113-045-7. [49]

Ph. Dutré and Y.D. Willems. Importance-driven Monte Carlo Light Tracing. In G. Sakas, P. Shirley, and S. Möller, editors, *Proceedings of the 5th Eurographics Workshop on Rendering, Darmstadt, Germany (June 1994).*, 1994. [9, 163]

Ph. Dutré and Y.D. Willems. Potential-driven Monte Carlo Particle Tracing for Diffuse Environments with Adaptive Probability Functions. In Pat Hanrahan and Werner Purgathofer, editors, *Proceedings of the 6th Eurographics Workshop on Rendering, Dublin, Ireland (June 1995)*, pages 306–315, 1995. [163]

Ph. Dutré, E.P. Lafortune, and Y.D. Willems. Monte Carlo Light Tracing with Direct Computation of Pixel Intensities. In H. P. Santo, editor, *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 128–137, 1993. [8, 109]

Philip Dutre. *Mathematical Frameworks and Monte Carlo Algorithms for Global Illumination in Computer Graphics*. PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, September 1996. [90, 108]

Shaohua Fan, Stephen Chenney, and Yu-Chi Lai. Metropolis Photon Sampling with Optional User Guidance. In *Rendering Techniques '05 (Proceedings of the*

16th Eurographics Symposium on Rendering), pages 127–138. Eurographics Association, 2005. [162]

Jeppe Revall Frisvad. *Light, Matter, and Geometry: The Cornerstones of Appearance Modelling.* PhD thesis, Technical University of Denmark, May 2008. [5]

A. Gelman, G.O. Roberts, and W.R. Gilks. Efficient Metropolis Jumping Rules. *Bayesian Statistics*, 5:599–607, 1995. [85]

Reid Gershbein, Peter Schröder, and Pat Hanrahan. Textures and Radiosity: Controlling Emission and Reflection with Texture Maps. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 51–58, New York, NY, USA, 1994. ACM. ISBN 0-89791-667-0. [51]

W.R. Gilks, S. Richardson, and David Spiegelhalter. *Markov Chain Monte Carlo in Practice.* Chapman & Hall/CRC, 1995. [77]

Andrew S. Glassner. *Principles of Digital Image Synthesis.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994. ISBN 1558602763. [16, 26, 67]

Michael Goesele, Xavier Granier, Wolfgang Heidrich, and Hans-Peter Seidel. Accurate Light Source Acquisition and Rendering. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 621–630, New York, NY, USA, 2003. ACM. ISBN 1-58113-709-5. [49]

Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the Interaction of Light Between Diffuse Surfaces. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 213–222, New York, NY, USA, 1984. ACM. ISBN 0-89791-138-5. [7, 107]

M.D. Grossberg and S.K. Nayar. What is the Space of Camera Response Functions? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume II, pages 602–609, Jun 2003. [52]

Jörg Haber, Marcus Magnor, and Hans-Peter Seidel. Physically-Based Simulation of Twilight Phenomena. *ACM Trans. Graph.*, 24(4):1353–1373, 2005. ISSN 0730-0301. [49]

Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive Photon Mapping. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–8, New York, NY, USA, 2008. ACM. [107]

J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods.* Methuen & Co Ltd, 1964. [56]

Vagn Lundsgaard Hansen. *Functional Analysis: Entering Hilbert Space*. World Scientific, 2006. ISBN 981-256-686-4(pbk). [90, 92]

W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970. [80, 82]

Eugene Hecht. *Optics*. Addison Wesley, 4th edition, 2002. ISBN 0-8053-8566-5. [33, 46]

Paul S. Heckbert. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 145–154, New York, NY, USA, 1990. ACM. ISBN 0-89791-344-2. [105]

L. G. Henyey and J. L. Greenstein. Diffuse Radiation in the Galaxy. *Astrophysics Journal*, 88:70–83, 1941. [48]

Heinrich Hey and Werner Purgathofer. Importance Sampling With Hemispherical Particle Footprints. In *SCCG '02: Proceedings of the 18th spring conference on Computer graphics*, pages 107–114, New York, NY, USA, 2002. ACM. ISBN 1-58113-608-0. [163]

David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A Radiosity Method for Non-Diffuse Environments. *SIGGRAPH Comput. Graph.*, 20(4): 133–142, 1986. ISSN 0097-8930. [7]

Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. Radiance Caching for Participating Media. *ACM Trans. Graph.*, 27(1):1–11, 2008a. ISSN 0730-0301. [164]

Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. Irradiance Gradients in the Presence of Participating Media and Occlusions. *Computer Graphics Forum (Proceedings of EGSR 2008)*, 27(4), 2008b. [164, 166]

Wojciech Jarosz, Nathan A. Carr, and Henrik Wann Jensen. Importance Sampling Spherical Harmonics. *Computer Graphics Forum (Proc. Eurographics EG'09)*, 28(2):577–586, 4 2009. [164, 165]

Henrik Wann Jensen. Importance Driven Path Tracing using the Photon Map. In *in Eurographics Rendering Workshop*, pages 326–335. Springer-Verlag, 1995. [9, 162]

Henrik Wann Jensen. Global Illumination using Photon Maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 21–30, London, UK, 1996. Springer-Verlag. ISBN 3-211-82883-4. [162]

Henrik Wann Jensen. *Realistic Image Synthesis using Photon Mapping*. AK Peters Ltd., 2001. [8, 107]

Henrik Wann Jensen and Per H. Christensen. Efficient Simulation of Light Transport in Scences with Participating Media using Photon Maps. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 311–320, New York, NY, USA, 1998. ACM. ISBN 0-89791-999-8. [162]

Henrik Wann Jensen, Frédo Durand, Julie Dorsey, Michael M. Stark, Peter Shirley, and Simon Premože. A Physically-Based Night Sky Model. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 399–408, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X. [49]

Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 2002. [146]

James T. Kajiya. The Rendering Equation. In *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, pages 143–150. ACM, 1986. [8, 26, 107]

Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo Methods, Volume I: Basics*. Wiley-Interscience Publications, 1986. [56, 63, 65, 66, 71, 72, 86]

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. In *Computer Graphics Forum*, pages 531–540, 2002. [115, 144, 153]

Alexander Keller. A Quasi-Monte Carlo Algorithm for the Global Illumination Problem in the Radiosity Setting. In *Monte-Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 239–251. Springer, 1995. [86]

Alexander Keller. Instant Radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. [8, 107]

Alexander Keller and Ingo Wald. Efficient Importance Sampling Techniques for the Photon Map. In *VMV*, pages 271–280, 2000. [162]

David Kirk and James Arvo. Unbiased Sampling Techniques for Image Synthesis. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 153–156, New York, NY, USA, 1991. ACM. ISBN 0-89791-436-8. [86]

Craig Kolb, Don Mitchell, and Pat Hanrahan. A Realistic Camera Model for Computer Graphics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. [52]

Thomas Kollig and Alexander Keller. Illumination in the Presence of Weak Singularities. In Harald Niederreiter and Denis Talay, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 245–257. Springer Berlin Heidelberg, 2004. [109, 113]

Jaroslav Křivánek, Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Improved Radiance Gradient Computation. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*, pages 155–159, New York, NY, USA, 2005a. ACM. ISBN 1-59593-203-6. [163, 166]

Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance Caching for Efficient Global Illumination Computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550–561, 2005b. ISSN 1077-2626. [163]

E. P. Lafortune and Y. D Willems. Bi-Directional Path Tracing. In H. P. Santo, editor, *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, 1993. [9, 111]

Eric P. Lafortune. *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering.* PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, February 1995. [90, 108]

Eric P. Lafortune and Yves D. Willems. Reducing the Number of Shadow Rays in Bidirectional Path Tracing. In *in Proceedings of WSCG 95, (Pilsen, Czech Republic*, pages 384–392, 1995a. [113]

Eric P. Lafortune and Yves D. Willems. A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In P.M. Hanrahan and W. Purgathofer, editors, *Proceedings of the Sixth Eurographics Workshop on Rendering, Dublin, Ireland (June 1995)*, pages 11–20, 1995b. [163]

Eric P. Lafortune and Yves D. Willems. Rendering Participating Media with Bidirectional Path Tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 91–100, London, UK, 1996. Springer-Verlag. ISBN 3-211-82883-4. [103, 111]

Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Adaptive Numerical Cumulative Distribution Functions for Efficient Importance Sampling. In *Eurographics Symposium on Rendering*, June 2005. [149]

Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Discontinuity Meshing for Accurate Radiosity. *IEEE Comput. Graph. Appl.*, 12(6):25–39, 1992. ISSN 0272-1716. [7]

A. W. Marshall. The Use of Multi-Stage Sampling Schemes in Monte Carlo Computations. In *Symposium on Monte Carlo Methods*, pages 123–140, New York, NY, USA, 1956. Wiley. [73]

Nicholas Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the Americal Statistical Association*, 44(247), 1949. [57]

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. [80, 82]

Don P. Mitchell. Consequences of Stratified Sampling in Graphics. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 277–280, New York, NY, USA, 1996. ACM. ISBN 0-89791-746-4. [71]

Don P. Mitchell and Arun N. Netravali. Reconstruction Filters in Computer Graphics. *SIGGRAPH Comput. Graph.*, 22(4):221–228, 1988. ISSN 0097-8930. [52]

R. Keith Morley, Solomon Boulos, Jared Johnson, David Edwards, Peter Shirley, Michael Ashikhmin, and Simon Premože. Image Synthesis using Adjoint Photons. In *GI '06: Proceedings of Graphics Interface 2006*, pages 179–186, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society. ISBN 1-56881-308-2. [109]

F. E. Nicodemus. Radiance. *American Journal of Physics*, 31(5):368–367, 1963. [30, 39]

F. E. Nicodemus. *Self-Study Manual on Optical Radiation Measurements: Part I—Concepts, Chapters 1 to 3.* Radiometric Physics Division, NIST, 1976. [26, 27]

F. E. Nicodemus. *Self-Study Manual on Optical Radiation Measurements: Part I—Concepts, Chapters 4 to 5.* Radiometric Physics Division, NIST, 1978. [35, 53]

F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. *Geometrical Considerations and Nomenclature for Reflectance.* Jones and Bartlett Publishers, Inc., USA, 1977. [26, 36, 42, 45]

Mark Pauly. Robust Monte Carlo Methods for Photorealistic Rendering of Volumetric Effects. Master's thesis, University of Kaiserslautern, 1999. [90, 95, 103, 113]

Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis Light Transport for Participating Media. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 11–22, London, UK, 2000. Springer-Verlag. ISBN 3-211-83535-0. [90, 95, 98, 113, 114]

Frank L. Pedrotti, Leno M. Pedrotti, and Leno S. Pedrotti. *Introduction to Optics*. Pearson Prentice HAll, 3rd edition, 2007. ISBN 0-13-197133-6. [33]

K. Perlin and E. M. Hoffert. Hypertexture. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 253–262, New York, NY, USA, 1989. ACM. ISBN 0-89791-312-4. [104]

Ingmar Peter and Georg Pietrek. Importance Driven Construction of Photon Maps. In *In Rendering Techniques 1998 (Proceedings of the 9th Eurographics Workshop on Rendering*, pages 269–280. Springer-Verlag, 1998. [162]

Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 012553180X. [11]

Bui Tuong Phong. Illumination for Computer Generated Pictures. *Commun. ACM*, 18(6):311–317, 1975. ISSN 0001-0782. [40]

Michael Potmesil and Indranil Chakravarty. A Lens and Aperture Camera Model for Synthetic Image Generation. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, pages 297–305, New York, NY, USA, 1981. ACM. ISBN 0-89791-045-1. [52]

A. J. Preetham, Peter Shirley, and Brian Smits. A Practical Analytic Model for Daylight. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 91–100, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. [49]

Rudolpj W. Preisendorfer. *Radiative Transfer on Discrete Spaces*. Pergamon Press, 1965. [16]

Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray Tracing on Programmable Graphics Hardware. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 703–712, New York, NY, USA, 2002. ACM. ISBN 1-58113-521-1. [9]

Matthias Raab, Daniel Seibert, and Alexander Keller. Unbiased Global Illumination with Participating Media. In Alexander Keller, Stefan Heinrich, and Harald Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 591–606. Springer Publishing Company, Incorporated, 2007. [104, 115, 144]

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2005. [77]

Holly E. Rushmeier and Kenneth E. Torrance. The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 293–302, New York, NY, USA, 1987. ACM. ISBN 0-89791-227-6. [7]

Mutsuo Saito and Makoto Matsumoto. SIMD-Oriented Fast Mersenne Twister: a 128-bit Pseudorandom Number Generator. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 607–622. Springer Berlin Heidelberg, 2006. [63]

Charles M. Schmidt and Brian Budge. Simple Nested Dielectrics in Ray Traced Images. *journal of graphics tools*, 7(2):1–8, 2002. [33]

Arthur Schuster. Radiation Through a Foggy Athmosphere. *Astrophysical Journal*, 21(1):1–22, 1905. [16]

K. Schwarzschild. On The Equilibrium of The Sun's Atmosphere. *Nachrichten von der Königlichen Gesellschaft der Wissenschaften zu Göttingen. Math.-phys Klasse*, 195:41–53, 1906. [16]

Peter S. Shirley. *Physically Based Lighting Calculations for Computer Graphics*. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1991. [86, 109]

Brian E. Smits, James R. Arvo, and David H. Salesin. An Importance-Driven Radiosity Algorithm. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, pages 273–282. ACM Press, 1992. [7]

John M. Snyder and Alan H. Barr. Ray Tracing Complex Models Containing Surface Tessellations. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 119–128, New York, NY, USA, 1987. ACM. ISBN 0-89791-227-6. [43]

Frank Suykens and Yves D. Willems. Density Control for Photon Maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 23–34, London, UK, 2000. Springer-Verlag. ISBN 3-211-83535-0. [162]

L. Szirmay-Kalos, P. Dornbach, and W. Purgathofer. On the Startup Bias Problem of Metropolis Sampling. Technical report, Department of Control Engineering and Information Technology, Technical University of Budapest, 1999. [114]

Eric Veach. Non-symmetric Scattering in Light Transport Algorithms. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 81–90, London, UK, 1996. Springer-Verlag. ISBN 3-211-82883-4. [38, 90]

Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997. [9, 25, 35, 39, 40, 41, 42, 46, 65, 70, 71, 75, 76, 90, 93, 94, 95, 101, 106, 112, 113, 116, 117, 119, 131, 143]

Eric Veach and Leonidas J. Guibas. Bidirectional Estimators for Light Transport. In *Eurographics Rendering Workshop 1994 Proceedings*, pages 147–162. Springer-Verlag, New York, 1994. [9, 95, 111]

Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. [9, 75, 112]

Eric Veach and Leonidas J. Guibas. Metropolis Light Transport. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. [9, 113, 114, 144, 157]

Channing Verbeck and Donald Greenberg. A Comprehensive Light-Source Description for Computer Graphics. *IEEE Comput. Graph. Appl.*, 4(7):66–75, 1984. ISSN 0272-1716. [50]

John von Neumann. Various Techniques Used in Connection with Random Digits. *J. Research Nat. Bur. Stand., Appl. Math. Series*, 12:36–38, 1951. [68]

Carsten Wächter. *Quasi-Monte Carlo Light Transport Simulation by Efficient Ray Tracing*. PhD thesis, Universität Ulm, 2007. [43]

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: A scalable approach to illumination. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1098–1107, New York, NY, USA, 2005. ACM. [107]

Greg Ward and Paul Heckbert. Irradiance Gradients. In *Eurographics Rendering Workshop*, pages 85–98, May 1992. [163]

Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A Ray Tracing Solution for Diffuse Interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92, New York, NY, USA, 1988. ACM. ISBN 0-89791-275-6. [163]

David R. Warn. Lighting Controls for Synthetic Images. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 13–21, New York, NY, USA, 1983. ACM. ISBN 0-89791-109-1. [50]

Turner Whitted. An Improved Illumination Model for Shaded Display. *Commun. ACM*, 23(6):343–349, 1980. ISSN 0001-0782. [7, 107]

Alexander Wilkie, Andrea Weidlich, Marcus Magnor, and Alan Chalmers. Predictive Rendering. In *SIGGRAPH ASIA '09: ACM SIGGRAPH ASIA 2009 Courses*, pages 1–428, New York, NY, USA, 2009. ACM. [3]

Andrew Woo, Andrew Pearce, and Marc Ouellette. It's Really Not a Rendering Bug, You See... *IEEE Comput. Graph. Appl.*, 16(5):21–25, 1996. ISSN 0272-1716. [43]

Günther Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae (Wiley Series in Pure and Applied Optics)*. Wiley-Interscience, 2 edition, 2000. ISBN 0471399183. [28, 31, 50]

Index